Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories

4-5 December 2009
Milan, Italy

Editors:
Marco Passarotti
Adam Przepiórkowski
Savina Raynaud
Frank Van Eynde

Published by
EDUCatt
Ente per il Diritto allo Studio Universitario dell'Università Cattolica
Largo Gemelli 1, 20123 Milano - tel. 02.72342235 - fax 02.80.53.215
e-mail: editoriale.dsu@unicatt.it (produzione); librario.dsu@unicatt.it (distribuzione)
web: www.unicatt.it/librario
ISBN: 978-88-8311-712-1

Cover illustration: a cloister of the Catholic University, Milan. Photo by Marco Passarotti.

Preface

The Eighth International Workshop on Treebanks and Linguistic Theories (TLT8) was held at the Catholic University of the Sacred Heart in Milan (Italy) on 4-5 December 2009 (see `http://tlt8.unicatt.it`). This was the first time that it has been held in Italy. Dates and locations of the previous workshops are provided in a separate section.

Since its first edition in 2002, TLT has provided a forum for discussion of methods and tools for the design, creation and exploitation of treebanks and the linguistic theories acting as their background. Today, treebanks are essential resources both for data-driven approaches to natural language processing and for linguistic research. Indeed, while treebank data are frequently exploited for tasks in computational linguistics such as grammar induction and the training of NLP tools, in linguistic research they can be used in order to refine and improve pre-corpus linguistic theories. Furthermore, large-scale data annotation allows for empirically evaluating the accuracy of a grammar and revising it on the basis of evidence.
Recently, many treebank projects for less-resourced languages have begun. The increasing spread of such treebanks benefits from the exploitation of tools and methods developed over the years for many similar projects for other languages. The language-independent status of these methods and tools has indeed allowed their re-usability (or easy adaptation) to many different languages. This has eased and sped up the process of creation and dissemination of treebanks for less-resourced languages. Another growing research direction is the development of parallel treebanks, which are vital resources for machine translation and comparative studies.

The call for papers for TLT8 requested for unpublished, completed work. 30 submissions were received, 25 for full papers, 5 for poster presentations. The submissions were authored by researchers from 19 different countries in America, Asia and Europe. Each submission was evaluated by three reviewers.
The Programme Committee consisted of 24 members (including the 4 co-chairs) from 14 different countries. They all worked as reviewers. Based on

their scores and the comments they provided on the content and quality of the papers, 15 papers and 4 posters were accepted for presentation and publication, which corresponds to an acceptance rate of 63.3%. The accepted submissions cover a wide range of topics related to both long-standing and new treebanks, reporting on aspects of their construction, querying, exploitation and evaluation. As requested in the call for papers, this edition puts a particular emphasis on projects aiming to compile representative treebanks for less-resourced, ancient and/or dead languages.

Completing the programme are the invited lectures by Roberto Busa SJ (Catholic University of the Sacred Heart, Milan) and Eva Hajičová (Charles University, Prague, Czech Republic). There is a connection between the research work of the two invited speakers thanks to the ongoing project of the *Index Thomisticus* Treebank (at the Catholic University in Milan), whose annotation guidelines were designed according to those of the Prague Dependency Treebank.

Following in the tradition of TLT's recent editions, a co-located event was also organised (see `http://tlt8.unicatt.it/framenet.htm`). This one-day event, preceding TLT8, was devoted to the FrameNet project and conceived as a masterclass and talk by Charles J. Fillmore (in the morning) followed in the afternoon by a workshop with seven oral presentations (peer-reviewed) on research concerning FrameNet and related linguistic and corpus topics. The organization of this co-located event arose from the consideration that, while the FrameNet team has begun to annotate some texts as a demonstration of how frame semantics can contribute to text understanding, no FrameNet-annotated large corpus is currently available, and FrameNet data are systematically biased by the criteria for the selection of the examples adopted to describe the frame semantics of target words. Therefore, a closer collaboration between FrameNet and annotated corpora (especially, treebanks) is now required for at least two reasons: (a) during the procedure of syntactic annotation, FrameNet data can help with consistency and can supply motives in support of annotation choices; (b) annotated corpora provide further evidence for FrameNet data, allowing lexicographers to ground their decisions on a wider variety of examples. Thus, the aim of this workshop was to put people who are involved in treebank development, management and exploitation into contact with the FrameNet project.

Words of gratitude and acknowledgement are due to the local organizers and hosts, the 'Servizio Formazione Permanente' and the CIRCSE Research Centre, which took the initiative of organizing TLT8.
We also gratefully mention the financial support received from Fondazione IBM Italia (`www.fondazioneibm.it`), Vega Informatica (`www.vega.it`), Banca Mediolanum (`www.bancamediolanum.it`) and DAMA (`www.ricamificiodama.it`; many thanks for the beautiful

merchandising). The TLT8 logo was designed by Nicola Tibiletti: we thank him very much.

TLT8 was endorsed by CLARIN (www.clarin.eu/) and FLaReNet (www.flarenet.eu/), and held under the patronage of the Departments of Philosophy, and Linguistic Sciences and Foreign Literatures at the Catholic University of the Sacred Heart.

The TLT8 Co-Chairs

Marco Passarotti, Catholic University of the Sacred Heart, Milan, Italy
Adam Przepiórkowski, Polish Academy of Sciences, Warsaw, Poland
Savina Raynaud, Catholic University of the Sacred Heart, Milan, Italy
Frank Van Eynde, University of Leuven, Belgium

Programme Committee

Organizing Committee

Proceedings of previous TLT workshops and invited speakers

E. Hinrichs & K. Simov (eds.), Proceedings of the First Workshop on Treebanks and Linguistic Theories, Sozopol (Bulgaria), September 20-21, 2002. v + 274 pages.
http://www.bultreebank.org/Proceedings.html.

J. Nivre & E. Hinrichs (eds.), Proceedings of the Second Workshop on Treebanks and Linguistic Theories, Växjö (Sweden), November 14-15, 2003. Växjö University Press. 232 pages.
Invited speakers: Thorsten Brants (Google Inc.), Stephan Oepen (U. of Oslo).

S. Kübler, J. Nivre, E. Hinrichs & H. Wunsch (eds.), Proceedings of the Third Workshop on Treebanks and Linguistic Theories, Tübingen (Germany), December 10-11, 2004. Eberhard Karls Universität Tübingen, Seminar für Sprachwissenschaft. vi + 203 pages.
Invited speakers: Collin Baker (ICSI, Berkeley), Fred Karlsson (U. of Helsinki).
Publication of selected papers in: E. Hinrichs & K. Simov (eds.), Treebanks and Linguistic Theories. Special Issue of the Journal on Research on Language and Computation. Vol. 2, Nr. 4, 2004.

M. Civit, S. Kübler & M.A. Martí (eds.), Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories, Barcelona (Spain), December 9-10, 2005. Universitat de Barcelona, Publicacions i Edicions. 220 pages.
Invited speakers: Frank van Eynde (U. Leuven), Manfred Pinkal (U. of the Saarland).

J. Hajič & J. Nivre (eds.), Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories, Prague (Czech Republic), December 1-2, 2006. Univerzita Karlova v Praze, Ústav Formální a Aplikované Lingvistiky. 258 pages.
Invited speakers: Gosse Bouma (U. of Groningen), Martha Palmer (U. of Colorado at Boulder).

K. De Smedt, J. Hajič & S. Kübler (eds.), Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories, Bergen (Norway), December 7-8, 2007. Northern European Association for Language Technology Proceedings Series, Vol. 1. viii + 218 pages.
`http://dspace.utlib.ee/dspace/handle/10062/4476.`
Invited speakers: Erhard Hinrichs (U. of Tübingen), Julia Hockenmaier (U. of Illinois).

F. van Eynde, A. Frank, K. De Smedt & G. van Noord (eds.), Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories, Groningen (The Netherlands), January 23-24, 2009. Landelijke Onderzoekschool Taalwetenschap, Occasional Series. 197 pages.
Invited speakers: Robert Malouf (San Diego State U.), Adam Przepiórkowski (Polish Academy of Sciences).

Contents

From Punched Cards to Treebanks: 60 Years of Computational Linguistics

Roberto Busa SJ
Catholic University of the Sacred Heart, Milan, Italy
E-mail: roberto.busa@unicatt.it

Abstract

I will cover some of the main stages of my projects in Computational Linguistics through 60 years, from their very beginning after World War II to the Internet era, including my most recent works on 'Lessico Tomistico Biculturale' and the syntactic annotation of data in the 'Index Thomisticus' Treebank project. The talk will describe how and why the idea of electronic data processing in literary and linguistic analysis came to my mind at first, remembering my meeting with Mr. Thomas Watson at IBM and his decision to fund the realization of 'Index Thomisticus' since 1949. I will describe how meticuolous is the research methodology in linguistics which is imposed by (and thanks to) the use of computers. Through the years, I applied this methodology to more than 20 different languages (starting from Latin) and to many fields of Computational Linguistics, such as semiautomatic lemmatization, machine dictionaries, textual typology, methods of treatment of different alphabets etc.

From Prague Structuralism to Treebank Annotation

Eva Hajičová
Charles University in Prague, Czech Republic
Institute of Formal and Applied Linguistics
E-mail: `eva.hajicova@mff.cuni.cz`

Abstract

When contemplating about the relationships between treebanking and linguistic theory, we believe it is useful to look a little bit further back and to consider (also) the roots of linguistic science rather than to restrict the attention to the theoretical aspects of what might be viewed as contemporary theoretical developments. In our contribution, we focus on three tenets of the structural and functional approach of the Prague School of Linguistics as reflected in the writings of Praguian linguists of the classical period, namely (i) the concept of markedness and the opposition of the core and periphery of language system, (ii) the functional viewpoint, and (iii) the frequently assumed distinction between the "grammatical" and "semantic" structure of the sentence, with the emphasis on the underlying syntactic relations with the verb as the nucleus. These three aspects are discussed both in their original, fundamental treatments as well as in view of the possibility and usefulness of their application to treebank annotation on different layers. In this connection, such questions are posed as what linguists (be they theoretically, computationally, or corpus- oriented) have expected to get from treebank annotation, in which respects these expectations have been met and in which we have failed or at least temporarily failed, which blind alleys we have fallen in on the way, which unexpected problems we have encountered. In the end, we will consider what we assume can(not) be done without treebanking (tree representations) and we will try to summarize what the treebanks are good for in spite of all the limits, insufficiencies or even (temporary) failures.

An Ownership Model of Annotation: The Ancient Greek Dependency Treebank

David Bamman, Francesco Mambrini and Gregory Crane
The Perseus Project, Tufts University

Abstract

We describe here the first release of the Ancient Greek Dependency Treebank (AGDT), a 190,903-word syntactically annotated corpus of literary texts including the works of Hesiod, Homer and Aeschylus. While the far larger works of Hesiod and Homer (142,705 words) have been annotated under a standard treebank production method of soliciting annotations from two independent reviewers and then reconciling their differences, we also put forth with Aeschylus (48,198 words) a new model of treebank production that draws on the methods of classical philology to take into account the personal responsibility of the annotator in the publication and ownership of a "scholarly" treebank.

1 Introduction

Data-driven research in linguistics relies on the existence of a large body of texts that have been annotated on several linguistic levels. For modern languages like English, these tend to be comprised of genres like newswire; for Latin, Greek, and other historical languages, our observations are based on a smaller but more heavily studied canon. An article from the *Wall Street Journal* is certainly more representative of how native English speakers actually speak than Homer's epic *Iliad* is for ancient Greeks, but the *Iliad* has been a focused object of study for almost 3,000 years, with schoolchildren and tenured professors alike scrutinizing its every word, annotating its syntax, semantics and other linguistic levels either privately in the margins of their books or as published commentaries.

Recent scholarship has seen the rise of a number of treebanks for historical languages over the past few years, including Middle English [11], Early Modern English [10], Old English [22], Medieval Portuguese [18], Ugaritic [24], Latin [1, 15] and several Indo-European translations of the New Testament [8]. The long history of philological research on the individual texts that constitute these works highlights what is perhaps the greatest difference between syntactically annotated corpora for modern languages and those for historical ones – while ambiguity is of course present in all language, the individual ad hoc decisions that annotators make in resolving syntactic ambiguity when creating modern treebanks have, for heavily studied Classical and other historical texts, been debated for centuries; dissertations and entire careers have been made on the study of a single work of a single author. Since over two thousand years separate us from the time when Greek and Roman authors were writing, Classical texts also have additional confounding factors which bring this debate to new levels – not simply the interpretation of the text as it appears to us, but what actually constitutes that text itself.

In creating an annotated corpus of a language for which no native speakers exist (and for which we subsequently cannot rely on native intuitions), we are building on a mountain of prior scholarship that has shaped our fundamental understanding of the text. In order to accommodate this level of scholarly debate on a basic level of annotation, we describe here a new mode of treebank production – what we are terming a "scholarly" treebank. Just as every critical edition and commentary bears the mark and reputation of its author, including the cultural context in which it was written, every act of annotation is here associated with the individual who created it. By stressing such ownership, we hope to transform the act of treebanking from an anonymous practice into a mode of scholarly publication.

The aim of this paper is twofold: we present the first release of the Ancient Greek Dependency Treebank (AGDT), containing 190,903 words of Ancient Greek annotated under a dependency grammar, and describe how it can form the core of scholarly treebanks to come.

2 The Ancient Greek Dependency Treebank

Ancient Greek is a highly inflected language with a considerable degree of variability in its word order. Even the comparatively simpler texts of Homer manifest a high degree of non-projectivity, where constituents themselves are broken up with elements of other constituents, as in the dependency graph shown in figure 1, where an arc drawn from μῆνιν ("rage") to Ἀχιλῆος ("Achilles") crosses that drawn from the root of the sentence to ἄειδε ("sing").[1] This flexibility has encouraged us to base our annotation style on the dependency grammar used by the Prague Dependency Treebank [6] for Czech (another non-projective language), which has since been widely adopted by a number of annotation projects for other languages, including Arabic [7] and Modern Greek [16]. Since Latin and Ancient Greek are so closely related, our specific guidelines have been built as an extension of those used for the Latin Dependency Treebank [1] and the Index Thomisticus [15].

2.1 Annotation

The efficient annotation of Ancient Greek is hindered both by the fact that no native speakers exist and that the texts we have available are typically highly stylized

[1]See Nivre [13] for a formal definition of projectivity.

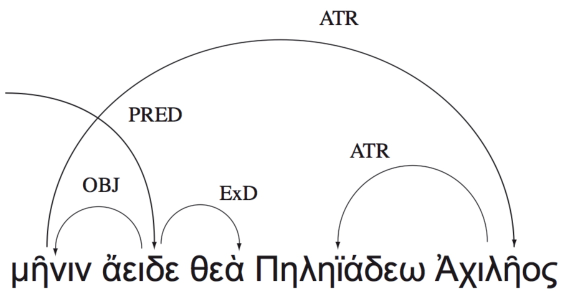


Figure 1: μῆνιν ἄειδε ϑεὰ Πηληϊάδεω Ἀχιλῆος ("Sing, goddess, of the rage of Achilles, the son of Peleus"), Homer, *Iliad* 1.1. Arcs are drawn from heads to their dependents.

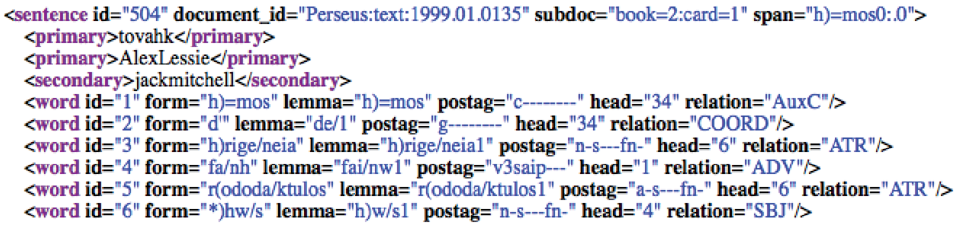
in nature. This difficulty and the ability of a sentence to present multiple valid syntactic interpretations has an impact on both annotation speed and inter-annotator agreement. While the Penn Treebank can report a productivity rate of between 750 and 1000 words per hour for their annotators after four months of training [21] and the Penn Chinese treebank can report a rate of 240-480 words per hour [3], our annotation speeds are significantly slower, ranging from 97 words per hour to 211, with an average of 124. Since we preserve the individual streams of annotation from all annotators, we can calculate inter-annotator accuracy (IAA) measures for the treebank in its entirety. Table 1 presents three such measures drawn from [5]: attachment score (ATT), label score (LAB) and labeled attachment score (LABATT), each one being the average annotator accuracy compared to the final corrected data. While our ATT of 87.4% approaches the 91.5% and 89.2% reported by the CATiB Arabic Treebank [5], our LAB and LABATT scores are lower, averaging 85.3% and 80.6%, respectively.

	ATT	LAB	LABATT
Hesiod, W&D	85.1%	85.9%	79.5%
Homer, Iliad	87.1%	83.2%	79.3%
Hesiod, Odyssey	87.5%	85.7%	80.9%
Total	**87.4%**	**85.3%**	**80.6%**

Table 1: Average inter-annotator accuracy in terms of attachment (ATT), label (LAB) and labeled attachment (LABATT) scores.

The backgrounds of the annotators range from advanced undergraduate students to recent PhDs and professors, with the majority being students in graduate programs in Classics. To help provide reading support for more efficient annotations, we have embedded our annotation interface within a larger digital library that presents the Greek source text to be annotated along with contextualizing secondary publications such as translations, commentaries, and references in dictionaries. In addition to an initial training period, annotators are actively engaged in new learning by means of an online forum in which they can ask questions of each other and of project editors; this allows them to be kept current on the most up-to-date codifications to the annotation guidelines while also helping bring new annotators up to speed. In the "standard" model of production, every sentence is annotated by two independent annotators and the differences are then reconciled by a third. This reconciliation (or "secondary" annotation as it is encoded in the XML release) is undertaken by a more experienced annotator/editor, typically a PhD with specialization in the particular subject area (such as Homer).

As figure 2 illustrates, all annotations are publicly released with the usernames of the primary and secondary annotators (which are then also associated with real names and institutional affiliations). By publicly acknowledging authorship, we are making our first steps toward an ownership model for annotation (more fully discussed below) and hope to provide a means for students, both graduate and undergraduate alike, to engage in the act of scholarly research and produce scientific data that can be useful to the wider Classics community.



```
<sentence id="504" document_id="Perseus:text:1999.01.0135" subdoc="book=2:card=1" span="h)=mos0:.0">
  <primary>tovahk</primary>
  <primary>AlexLessie</primary>
  <secondary>jackmitchell</secondary>
  <word id="1" form="h)=mos" lemma="h)=mos" postag="c--------" head="34" relation="AuxC"/>
  <word id="2" form="d'" lemma="de/1" postag="g--------" head="34" relation="COORD"/>
  <word id="3" form="h)rige/neia" lemma="h)rige/neia1" postag="n-s---fn-" head="6" relation="ATR"/>
  <word id="4" form="fa/nh" lemma="fai/nw1" postag="v3saip---" head="1" relation="ADV"/>
  <word id="5" form="r(ododa/ktulos" lemma="r(ododa/ktulos1" postag="a-s---fn-" head="6" relation="ATR"/>
  <word id="6" form="*)hw/s" lemma="h)w/s1" postag="n-s---fn-" head="4" relation="SBJ"/>
```

Figure 2: XML fragment from the AGDT (Homer, *Odyssey* 2.1).

2.2 AGDT 1.0

Using this model, we have annotated a total of 190,903 words from three different authors (Hesiod, Homer and Aeschylus), as distributed in table 2.

In addition to the index of its syntactic head and the type of relation to it, each word is also annotated with the lemma from which it is inflected and its morphological code (a composite of nine different morphological features: part of speech, person, number, tense, mood, voice, gender, case and degree). All of the files have been freely released under a Creative Commons license.[2]

For the works of Homer and Hesiod, we have followed the standard production method of soliciting annotations from two different annotators and then reconciling the differences between them. Aeschylus, whose textual tradition is much more fragmentary, has presented an ideal case for annotation as a scholarly treebank.

[2]All treebank data can be found at: `http://nlp.perseus.tufts.edu/syntax/treebank/`.

Method	Author	Work	Sentences	Words
Standard	Hesiod	Works and Days	446	6,214
	Homer	Iliad	2,470	37,223
		Odyssey	6,417	99,268
Scholarly	Aeschylus	Agamemnon	809	9,796
		Eumenides	521	6,376
		Libation Bearers	572	6,563
		Persians	478	6,223
		Prometheus Bound	589	7,045
		Seven Against Thebes	478	6,206
		Suppliants	518	5,989
		Total:	**13,298**	**190,903**

Table 2: AGDT 1.0 composition by work.

3 Scholarly Treebanks

Linguistic annotation projects have, of necessity, long focused on the creation of the single-best annotation, enforcing agreement between annotators even in cases of ambiguity. This approach works well for generic text such as newswire (where the value lies not in any individual sentence but rather in the aggregation of many) but breaks down when the objects of annotation are themselves the focus of scholarly debate. In these cases we must provide a means for encoding multiple annotations for a text and allowing scholars who disagree with a specific annotation to encode their disagreement in a quantifiable form.

For historical texts especially, scholarly disagreement can be found not only on the level of the correct syntactic parse, but also on the form of the text itself. These two levels are not completely isolated from each other, since it is often a scholar's understanding of the meaning of the text – i.e., what it *should* say – that informs their decisions about its reconstruction (i.e., what it actually *did* say). The need for this reconstruction is due to the process of textual transmission. We do not have a copy of Plato's *Apology* in his own hand; what we have instead is a series of manuscripts, one copied from the other, with errors introduced into each generation by the process of hand-copying by medieval scribes. This manuscript transmission allowed the work of the author to survive, but resulted in a considerable alteration of the text. Modern critical editions attempt to reconstruct the original text by a systematic comparison of that manuscript tradition.

As the product of scholarly labor, a critical edition displays the text as it is reconstructed by an editor; it is thus an interpretative hypothesis whose foundations lie on the methods of textual criticism. A scholarly treebank may be defined by analogy as a syntactically annotated corpus that again reflects an interpretation of a single scholar, based not only on the scholar's philological acumen but also on an inevitable degree of personal taste and opinions that are culturally and historically determined. A scholarly treebank thus distances itself from the notion that linguistic annotations can be absolute; when dealing with non-native historical languages especially, a syntactic interpretation of a sentence is always the interpretation of an individual and therefore subject to debate.

3.1 Aeschylus

We have decided to treat the corpus of Aeschylus' plays as the first example of a scholarly treebank due to the difficulty (even by Classical standards) of its textual tradition. The historical position of this author (ca. 525 BCE – ca. 456 BCE) may partly account for this complexity. Classical authors established him as the true founder of tragedy,[3] the poet who took a genre that was already characterized by a high degree of linguistic diversity and complexity and transformed it from its humble and rustic origins into a sublime form of poetry.

When Ancient Greek literature was rediscovered in Western Europe in the Renaissance, the difficulty of reading Aeschylus (along with all other Classical texts) was increased by the errors that inevitably intruded into the text during the process of copying. Out of a whole corpus that included between 70 and 90 tragedies, a canon of seven plays traditionally attributed to Aeschylus was chosen most likely in late antiquity to be copied integrally (a number of fragments of other works also survived independently).[4] Of these, only three (*Prometheus Bound*, *Persians* and *Seven Against Thebes*) have been preserved by a group of manuscripts large enough to assure a good transmission, and two (*Libation Bearers* and *Suppliants*) survive only in one single manuscript (and its copies), the Laurentianus 32.9. Starting from this controversial evidence, a vast number of scholars, beginning from the first printed edition of 1518, have undertaken the enterprise of giving justice to the complex poetry of the author and amending the text of all errors [12]. The main bibliographic catalogue for Aeschylus lists no less than 127 editions of the seven plays for the years 1518-1974, counting also the major reissues [23, 633-35]; if we include the separate editions of the single tragedies or of the trilogy (the *Oresteia*), the count rises exponentially.

3.2 Example: *Agamemnon* 176-8

One example (*Ag*. 176-8) may give an idea of how this complex history affects the practical task of treebanking. In a pivotal passage of the so-called "Hymn to Zeus" in the *Agamemnon*, the chorus voices for the first time in the play a theological vision that will dominate the whole *Oresteia*: the rule of "learning through suffering" as the means by which Zeus instructs the mortals to wisdom. Smyth's

[3]Cf. Dioscorides (3rd century BCE), *Tragicorum Graecorum Fragmenta*, Testimonium 163 [17, 107-8].

[4]The conflicting ancient evidence on the number of plays attributed to Aeschylus in antiquity is collected by Radt [17].

[20] edition[5] of the Greek text reads:

> τὸν φρονεῖν βροτοὺς ὁδώ-
> [the . to be wise . mortals . putting on ...]
> σαντα, τὸν πάθει μάθος
> [... the way . the . through suffering . learning]
> θέντα κυρίως ἔχειν.
> [establishing . authoritatively . hold]

The precise meaning of the passage is subject to debate (see below), but a basic translation is: "[Zeus] ... who put men on the path of wisdom, who established that the law 'learning through suffering' shall be in force."

Though the formula πάθει μάθος ("learning through suffering") is both quoted and commented upon in many general introductions to the theater of Aeschylus (it was even quoted by Robert F. Kennedy in his speech on the assassination of Martin Luther King Jr. [9]), both the text and syntactic interpretation of the sentence are highly controversial.

For instance, we may note at once that the second masculine accusative article τὸν (l. 177: τὸν ... θέντα, "the one establishing") is a modern conjecture proposed by Schütz [19] and subsequently accepted by many followers, including Smyth above [20]. In contrast, all the manuscripts of the *Agamemnon* unanimously read a dative neuter article (τῷ) that is morphologically licensed to modify the dative noun πάθει instead ("the suffering").

This conjecture of Schütz is directly related to a question of syntactic interpretation. There is a fundamental ambiguity in the relationship between the two participles ὁδώσαντα ("put on the way") and θέντα ("establish"). Is it apposition (Zeus is the god that "put the men on the path of wisdom, *i.e.,* the one who established the law") or subordination ("Zeus gave wisdom to men *by establishing* the law")? Whichever interpretation we choose, the article with πάθει is rather difficult to understand. This inherent ambiguity on several levels has led the three most recent commentaries on the play – Fraenkel [4], Denniston-Page [14] and Bollack [2] – to adopt three very different solutions based on their own weighing of the philological evidence, each resulting in a markedly different syntactic tree. Figures 3 and 4 present these three trees annotated under a dependency grammar, and illustrate the variety of interpretations that have been argued in print for just this one sentence alone.[6]

[5] A digital version of this edition is available at: `http://www.perseus.tufts.edu/hopper/`.

[6] The different interpretations are, of course, reflected also in different translations or paraphrases. Fraenkel (1950): "it is Zeus who has put men on the way to wisdom by establishing as a valid law *By suffering they shall win understanding*" [4]; Denniston-Page (1957): "he who set men on the path to understanding, who laid down the law, 'learning through suffering', to hold good" [14]; Bollack (1981): "de celui qui a ouvert aux mortels le penser, posant qu'ils tiendraient principalement leur savoir par la souffrance" ("of the one who opened the way of thinking for mortals, by establishing that chiefly by their suffering they will have their knowledge") [2].

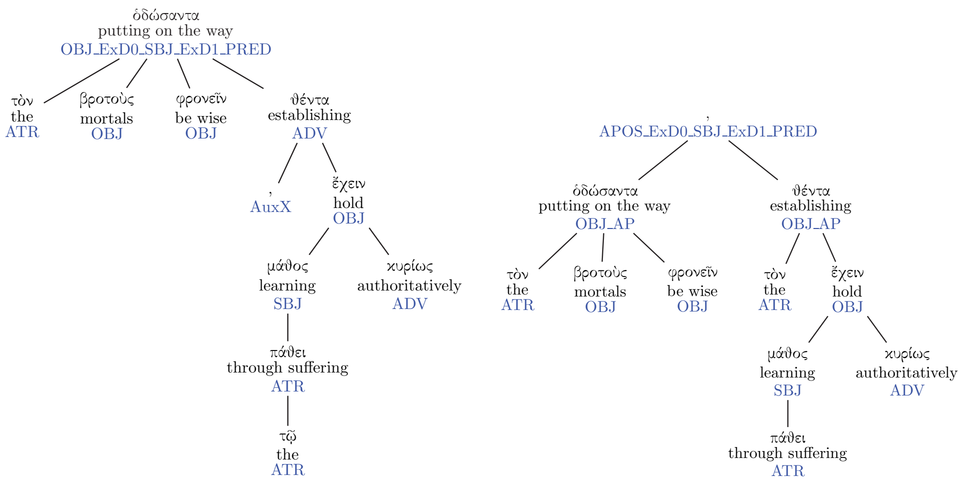


Figure 3: Trees of Fraenkel (left) and Denniston-Page (right) for *Ag.* 176-8.

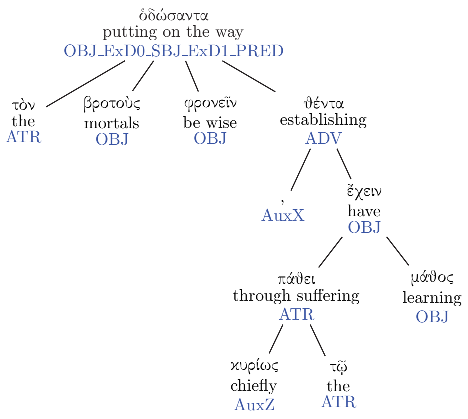
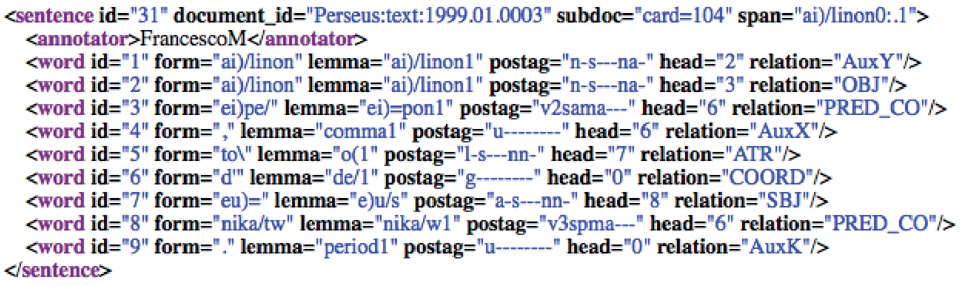


Figure 4: Bollack's tree for *Ag.* 176-8.

3.3 A Critical Treebank of Aeschylus

The variety of textual and syntactic interpretations for just these three lines of Aeschylus begins to point out the shortcomings of a standard treebank production model for texts of ongoing scholarly debate. While distributing the task of annotation across two independent annotators and then reconciling their differences does help remove any single annotator's personal bias from the final annotated corpus, for these texts what we want is exactly that – the quantified decisions of a single individual (whether Fraenkel, Denniston-Page, Bollack, or some other scholar), along with the sense of ownership and personal responsibly that attend such work. In this, the scholarly practice of annotation is practically indistinguishable from the creation of a critical edition of a text and attendant commentary.

For the complete works of Aeschylus, we have created a treebank based on the work of a single scholar following these philological principles. As in the creation of critical editions of texts, each syntactic annotation is created in consultation with the current state of Aeschlyean criticism; the resulting work stands as a contribution to that ongoing body of research. In total, the scholarly treebank amounts to 48,198 words (3,965 sentences) from 7 different plays and is included in the public release of AGDT 1.0 (see Figure 2). Figure 5 displays a fragment of that data – unlike the canonically produced texts of Homer and Hesiod, where a consensus is established among three individuals, this work here is the sole responsibility of the scholar who created it and remains that scholar's published interpretation of the text.



```
<sentence id="31" document_id="Perseus:text:1999.01.0003" subdoc="card=104" span="ai)/linon0:.1">
  <annotator>FrancescoM</annotator>
  <word id="1" form="ai)/linon" lemma="ai)/linon1" postag="n-s---na-" head="2" relation="AuxY"/>
  <word id="2" form="ai)/linon" lemma="ai)/linon1" postag="n-s---na-" head="3" relation="OBJ"/>
  <word id="3" form="ei)pe/" lemma="ei)=pon1" postag="v2sama---" head="6" relation="PRED_CO"/>
  <word id="4" form="," lemma="comma1" postag="u--------" head="6" relation="AuxX"/>
  <word id="5" form="to\" lemma="o(1" postag="l-s---nn-" head="7" relation="ATR"/>
  <word id="6" form="d'" lemma="de/1" postag="g--------" head="0" relation="COORD"/>
  <word id="7" form="eu)=" lemma="e)u/s" postag="a-s---nn-" head="8" relation="SBJ"/>
  <word id="8" form="nika/tw" lemma="nika/w1" postag="v3spma---" head="6" relation="PRED_CO"/>
  <word id="9" form="." lemma="period1" postag="u--------" head="0" relation="AuxK"/>
</sentence>
```

Figure 5: XML fragment from the AGDT (Aeschylus, *Ag.* 121).

4 Conclusion

By focusing on authorship in the release of the AGDT, we hope to drive future research in two directions. First, by publicly releasing the data with citable attributions of ownership, we hope to provide the core around which other interpretations of the data can be layered – a scholar who disagrees with a single annotation decision need not start from scratch to contribute a new annotation, but can simply build on the existing data and change only the elements subject to debate. As the example from *Agamemnon* 176-8 from above clearly shows, Classical texts very often license multiple syntactic interpretations, and providing a quantified record of how these multiple interpretations differ can only help drive future research.

Second, by publicly acknowledging the creator of the annotation, we hope to promote the act of treebanking as a scholarly publication no different than a critical edition or commentary. In so doing, we hope to engage a much wider audience in the creation of syntactically annotated data for historical languages – not only the corpus and computational linguists who have typically promoted them, but Classicists as well, for whom treebanking is simply a quantified form of the traditional scholarship that has been conducted for centuries.

5 Acknowledgments

Grants from the Alpheios Project (“Building a Greek Treebank”), the Andrew W. Mellon Foundation (“The CyberEdition Project: Workflow for Textual Data in Cyberinfrastructure”) and the National Endowment for the Humanities (PR-50013-08, “The Dynamic Lexicon: Cyberinfrastructure and the Automated Analysis of Historical Languages”) provided support for this work.

References

[1] David Bamman and Gregory Crane. The Latin Dependency Treebank in a cultural heritage digital library. In *Proceedings of the Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2007)*, pages 33–40, Prague, 2007. Association for Computational Linguistics.

[2] Jean Bollack and Pierre Judet de La Combe. *L’Agamemnon d’Eschyle: le texte et ses interprétations*. Presses universitaires de Lille, Lille, 1981.

[3] Fu-Dong Chiou, David Chiang, and Martha Palmer. Facilitating treebank annotation using a statistical parser. In *Proceedings of the First International Conference on Human Language Technology Research HLT ’01*, 2001.

[4] Eduard Fraenkel. *Aeschylus. Agamemnon*. Clarendon Press, Oxford, 1950.

[5] Nizar Habash and Ryan Roth. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore, 2009. Association for Computational Linguistics.

[6] J. Hajič. Building a syntactically annotated corpus: The Prague Dependency Treebank. In E. Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*. Prague, Charles University Press, 1998.

[7] J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, 2004.

[8] D.T.T. Haug and M.L. Jøhndal. Creating a Parallel Treebank of the Old Indo-European Bible Translations. In *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*, 2008.

[9] Robert F. Kennedy. Statement on the assassination of Martin Luther King, Indianapolis, Indiana, April 4, 1968.

[10] A. Kroch, B. Santorini, and L. Delfs. Penn-Helsinki Parsed Corpus of Early Modern English. http://www.ling.upenn.edu/hist-corpora/ppceme-release-1, 2004.

[11] A. Kroch and A. Taylor. Penn-Helsinki Parsed Corpus of Middle English, 2nd edition. http://www.ling.upenn.edu/hist-corpora/ppcme2-release-2/, 2000.

[12] Monique Mund-Dopchie. *La survie d'Eschyle à la Renaissance. Éditions, traductions, commentaires et imitations.* Peeters, Louvain, 1984.

[13] Joakim Nivre. Constraints on non-projective dependency parsing. In *EACL*. The Association for Computer Linguistics, 2006.

[14] Denys Page. *Aeschylus Agamemnon. Edited by the late John Dewar Denniston and Denys Page.* Clarendon Press, Oxford, 1957.

[15] Marco Passarotti. Verso il Lessico Tomistico Biculturale. La treebank dell'Index Thomisticus. In P. Raffaella and F. Diego, editors, *Il filo del discorso. Intrecci testuali, articolazioni linguistiche, composizioni logiche*, pages 187–205. Roma, Aracne Editrice, 2007.

[16] Prokopis Prokopidis, Elina Desipri, Maria Koutsombogera, Harris Papageorgiou, and Stelios Piperidis. Theoretical and practical issues in the construction of a Greek dependency treebank. In *In Proceedings of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160, 2005.

[17] Stefan Radt. *Tragicorum Graecorum Fragmenta. Vol. III: Aeschylus.* Vandehoeck und Ruprecht, Göttingen, 1985.

[18] Vitor Rocio, Mário Amado Alves, J. Gabriel Lopes, Maria Francisca Xavier, and Graça Vicente. Automated creation of a Medieval Portuguese partial treebank. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 211–227. Kluwer Academic Publishers, 2003.

[19] Christian Gottfried Schütz. *Commentationum in Aeschyli Agamemnon libellus I.* Jena, 1780.

[20] Herbert Weir Smyth. *Aeschylus. With an English Tranlsation.* Loeb Classical Library. Harvard University Press, Cambridge, 1922.

[21] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The Penn Treebank: An overview. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 5–22. Kluwer Academic Publishers, 2003.

[22] Ann Taylor, Anthony Warner, Susan Pintzuk, and Frank Beths. York-Toronto-Helsinki Parsed Corpus of Old English Prose, 2003.

[23] André Wartelle. *Bibliographie d'Eschyle et de la tragédie grecque 1518-1974.* Les Belles Lettres, Paris, 1978.

[24] Petr Zemánek. A treebank of Ugaritic: Annotating fragmentary attested languages. In *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT2007)*, pages 213–218, Bergen, 2007.

Finalising Multiword Annotations in PDT

Eduard Bejček and Pavel Straňák and Jan Hajič

Charles University in Prague, Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 118 00 Praha, Czech Republic
E-mail: `{bejcek,stranak,hajic}@ufal.mff.cuni.cz`

Abstract

We describe the annotation of multiword expressions and multiword named entities in the Prague Dependency Treebank. This paper includes some statistics of data and inter-annotator agreement. We also present an easy way to search and view the annotation, even if it is closely connected with deep syntactic treebank.

1 Introduction

The units of tectogrammatical layer of the Prague Dependency treebank should not be just words but lexemes. Essential for this improvement is annotation of multiword expressions. Its goal is to identify the multiword lexemes that should become single tectogrammatical nodes in future.

As there was a lack of such annotated data, the project Bejček, Straňák, and Schlesinger (2008) started four years ago. In this project (usually) two annotators have been reading the newspaper texts from PDT 2.0 (see Hajič et al., 2006), searching for multiword expressions in it, and annotating them. They have concentrated on both multiword named entities (NEs) and multiword lexemes. The aim of the project was to develop reliable training data for further research and to improve the state of "t-nodes" in the trees of PDT (see below in Section 1.1 about PDT). As a side effect, the lexicon of multiword lexemes was created, entries have been inserted into it and existing ones have been corrected throughout the whole annotation process.

Now the project is near the end. What was our approach, what are the results, are there any interesting outputs?

The paper starts with a few words about the corpus we used and the existing annotation we both use and enhance. The second section is about our annotation and the approach we chose. Section 3 brings overview of the annotation done, various types of inter-annotator agreement etc. At the end we report on the technical aspects of releasing the data, and how it can be used, which is work in progress.

1.1 PDT

The Prague Dependency Treebank 2.0 (Hajič, 2005) includes rich annotation on a deep syntactic level of almost 50,000 sentences (for details on this “tectogrammatical layer” see Mikulová et al., 2006). On this layer, each node should correspond to one lexeme, but this is not the case now: multiword expressions (MWEs) are still represented by several nodes each.

As mentioned above, our goal is to integrate MWEs like “New York City”, “computational linguistics” or “kick the bucket” each into one joint node in a syntactic tree (with a meaning “NE: place”, “lexicon entry: 39485 [gloss: science branch]” and “lexicon entry: 13985 [synonym: die]”, respectively).

The advantage of the existing syntactic annotation of our source text was the possibility of preannotation and consistency checking. That could be done because all instances of the same MWE should have the same tree structure.[1]

2 The Way of Annotation

In this section, we introduce very briefly our project. Much more information can be found in Bejček and Straňák (2009).

Both multiword NEs and lexemes – if multiword – are called MWEs in this paper. Assigning the type of the MWE itself (such as a particular lexicon entry) is of little importance comparing to the fact, that it is found and its boundaries are marked. To create a typology of NEs (or phrasemes) is not our aim. Our effort leads to simple annotation guidelines for annotators; concrete labels are just an aid for further classification. Thus we adopt nine main types of NEs from Ševčíková et al. (2007)[2] (such as person name, name of a place, address etc.) and use more than 5,000 lexicon entries for other lexemes (such as phrasemes, non-compositional or non-substitutable collocations).[3] These lexicon entries have been collected from three lexicons and the set has been extended by annotators. Thereby the lexicon called SemLex was developed.

For the majority of time, we had two annotators, who annotated the same texts in parallel. (Overall, we had five annotators during the time as it can be seen in Table 2, but that is not crucial.) However, when we had enough data for inter-annotator agreement evaluation, we stopped parallel annotations (only with an occasional testing parallel document). The amount of data annotated in parallel can be seen in Table 1. By now, 85 % of the whole data is annotated.

[1]There are some marginal cases where the structure is not exactly the same. These differences will hopefully disappear in future, perhaps in PDT 3.0.

[2]This typology (with embedded types) was used for manual annotation of a corpus (Kravalová et al., 2009).

[3]The main simple criterion was “principle of compositionality”—whether it could be disassembled to parts that compose the meaning; if not, it should be in the lexicon. The second one is “substitutability of a part”—i.e. the possibility to substitute its component words with synonyms. Then we have annotation guidelines and meetings, where problems are solved.

amount of parallel annotations	in nodes	in % of PDT
three annotators	464	14.7 %
two annotators	1201	38.1 %
one annotator	1044	33.1 %
total	2709	85.8 %
total by at least two annotators	1665	52.8 %
PDT t-layer	3156	100 %

Table 1: The amount of single, double and triple parallel annotations.

For preannotation of the text we use Czech_geo_named_ent_recognizer and Czech_named_ent_SVM_recognizer from the Tecto-MT framework (Žabokrtský et al., 2008). These find some NEs. We also use external preannotation of phrasemes provided by our colleague (see Hnátková (2002)).

2.1 GUI

We developed a tool for our task, a GUI for annotators. Although we actually annotate the nodes in the trees in the background, we need to show only plain sentence to annotators. For each syntactic tree, the surface sentence is generated and every annotator's operation on it is converted back into the nodes and saved. In addition, as the subtree forming just annotated expression is identified, all the other occurrences of that subtree can be found in the neighbourhood automatically. That assists annotators with their manual work. Annotators are also allowed to view, modify and extend the SemLex in the same GUI.

This tool can be used for any annotations of treebank, where the annotated trees are better viewed as plain text.

2.2 Merging SemLexes

An annotator works off-line. Also their SemLex is modified off-line. Therefore the longer the annotation proceeds the more their SemLexes differ. Then we need to merge SemLexes and return the new one back to them for further annotation. The process described bellow is for two annotators.

First, we merge all entries that could be merged automatically. That means either entries that were the same in both SemLexes, or entries that were inserted into just one of the SemLexes.

Second, conflicting entries are delegated to third annotator who decides the correct forms. For this task we use a modal editor.[4] That means an editor with one mode for typing a text (which we disable) and other for macro executing. Macro invocation could be very simple—we use only one key for each operation. These are the reasons why we use a modal editor for manual merging.

[4]Vim in our case, http://www.vim.org/.

We put all conflicting lexicon entries into a file, each as a simple list of its values, and provided a syntax highlighting for the editor. Where there are more values (i.e. a conflict), we show all of them in a warning colour. We prepare some simple macros for the editor, such as “go to next conflict”, “choose first value”, “insert a comment” etc. and disable the option of typing in the text. When the conflict is resolved, the warning colour disappears. It confirmed that it is very fast to create and very simple and safe to use.

Third, the decisions from the third annotator were imported back into the SemLex and added to merged entries from the first step. After that, the annotated data are modified to correspond with the new SemLex.

3 Statistics

All statistics presented in this section are calculated for all our users. They didn’t annotate the same data, though, as can be seen in Table 2. This table also shows the ratio of PDT annotated by each of them.

annotator\part	PDT	amount
#1	● ●	2.7 %
#2	●●●●●●●● ●●●●●●●●	55.0 %
#3	●●●●●●●●●●●●●●● ●●● ● ●●	67.2 %
#4	●● ● ●●● ● ●●	21.2 %
#5	●●●● ●●●	13.4 %

Table 2: Annotated parts and the ratio to the whole PDT per each annotator.

The annotated parts of PDT slightly differ,[5] but the overall characteristic stays. There is very similar usage of NEs across all annotators[6] in the Table 3.

Besides nine types of NEs, the annotators use approximately 8,000 of SemLex entries; some of them 100×,[7] third of them only once. Since there is no straight borderline stating whether an occurrence is a NE or shether it should be marked as a SemLex entry, the agreement has to be evaluated together for NEs and SemLex entries.

[5]For example after a changeover from one newspaper to another.

[6]Only annotator #1 evidently differs, but this one annotated only less than 3 % of PDT.

[7]Foremost lexicon entries are “state budget”, “annual meeting”, “environment”, “join stock company” etc.

annotator	#1	#2	#3	#4	#5
address	-	0.4 %	0.1 %	0.6 %	0.7 %
biblio	-	0.1 %	0.2 %	-	0.0 %
foreign	0.2 %	0.7 %	0.5 %	1.0 %	0.6 %
institution	9.7 %	22.6 %	19.4 %	24.1 %	21.7 %
location	6.2 %	6.1 %	8.4 %	8.5 %	12.3 %
object	30.6 %	10.3 %	14.2 %	16.1 %	14.7 %
other	3.2 %	13.1 %	16.3 %	10.9 %	15.6 %
person	38.2 %	30.9 %	32.0 %	30.9 %	26.5 %
time	12.1 %	15.9 %	8.8 %	7.8 %	7.8 %
All NEs	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %

Table 3: Usage of named entities by particular annotators

3.1 Weighted Kappa Agreement

Because of the complicated character of our annotations,[8] we use weighted kappa for inter-annotator agreement in Table 4.

annotators	#1	#2	#3	#4	#5
#1	*	0.61	-	-	-
#2	0.61	*	0.56	0.21	-
#3	-	0.56	*	0.55	0.73
#4	-	0.21	0.55	*	0.70
#5	-	-	0.73	0.70	*

Table 4: Pairwise weighted kappa.

3.2 Agreement with "`is_name_of_person`" in PDT

There is an attribute called `is_name_of_person` in PDT, which is though assigned automatically. Figure 5 shows the agreement with our annotation. The corresponding value should be "named entity: person" and it is in 86 % of nodes.

There are three reasons, why the value does not correspond:

- The name of the person is not multiword. The connection of two `is_name_of_persons` was not marked in PDT, therefore the dependency edge between two such nodes may or may not create a MWEs; they could be names

[8]Firstly, the annotations are assigned to nodes, which are in m:n mapping to words. Secondly, two annotators happen to annotate differently although their annotations have non-empty intersection. Thirdly, some MWEs fall into more than one category, like phraseme and a name of an institution at the same time.

[9]Such node was annotated, but not as "person". The name of person could be part of a name of institution or even part of some multiword lexeme.

Annotators	PDT	Ratio
t-node annotated as		
NE “person”	nothing	4.5 %
other[9]	`is_name_of_person`	2.3 %
nothing	`is_name_of_person`	6.9 %
NE “person”	`is_name_of_person`	86.3 %

Table 5: The agreement between the PDT attribute `is_name_of_person` and named entity “person”.

of two people as well. One such example is “Pucciniho Turandot” (Puccini’s Turandot), which is the name of the composer and the name of the title character of his opera.

- The name of the person is a part of another MWEs. In that case, only the larger one should be annotated. For example, there is a name of person in “Pěvecký recitál Petera Dvorského” (Peter Dvorský’s Choral Recital), but it is an object as a whole.
- There is a mistake in our annotation (or in PDT, theoretically).

There are some others annotations in PDT (namely FPHR, DPHR, IDPH, and CPHR[10]), which are significant for us. These disagreements will be checked and mistakes will be repaired—either automatically, or manually.

4 Publication of the Data

As PDT is stored in the PML format (Pajas and Štěpánek, 2005), we also use PML for our annotations. That allows us to store the data as a stand-off annotation in so called “s-files” separately from the rest of PDT annotations. The MWEs found in each document are saved in a file linked with other files in PDT containing all the other annotations of this document as well as the surface sentences. (See Figure 1.)

This format allows us to show annotations in editor and viewer TrEd (Pajas and Štěpánek, 2008). It can present any part of the annotation in easily comprehensible, uncluttered way. User may choose to show or hide many detailed information about every word in the sentence (or node in the tree). There is also PML-TQ extension in TrEd, which allows a user to ask the queries in an easy user-friendly way. The queries are translated into SQL and evaluated by the database server containing treebanks. The resulting trees can be displayed either in TrEd, or on the web page using SVG. An example of a query is in Figure 2. Another query (in Figure 3) combines more layers of annotation.

Our complete data will be released under PDT licence within several months.

[10]These are “foreign phrase”, “dependent part of a phraseme”, “identification structure”, and “copula verbonominal predicate”, respectively.

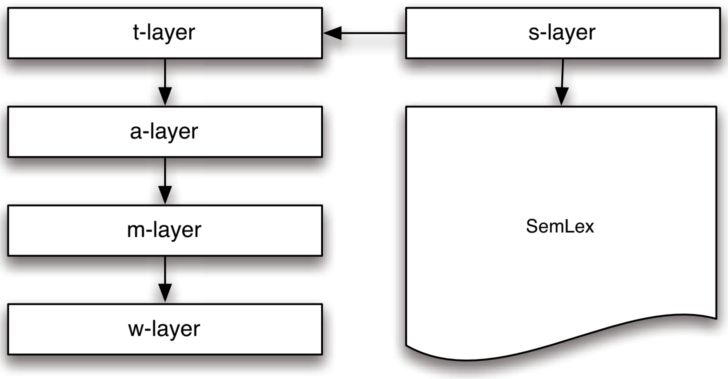
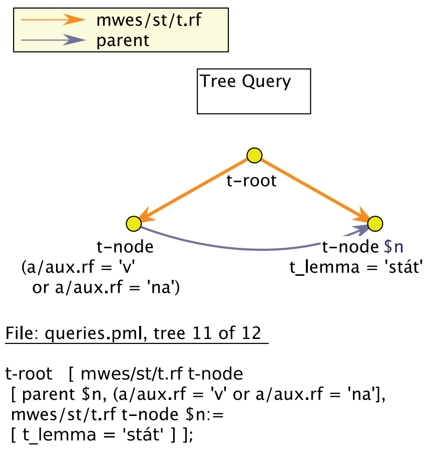


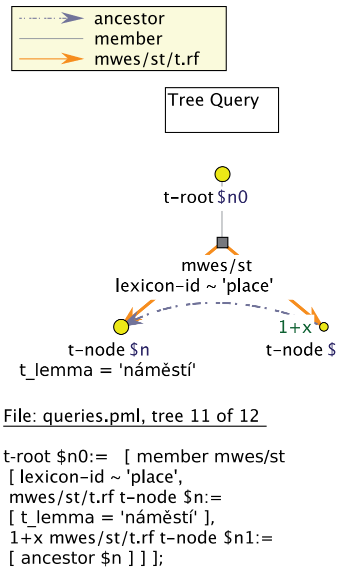
Figure 1: One document is stored in five interlinked files with links also to SemLex.

We will release the s-files themselves, which are very simple XML files, as well as scripts to merge the s-files with PDT to enable searching via PML-TQ and displaying the trees in TrEd. Export to CoNLL format (Hajič et al., 2009) will be also provided. CoNLL format is a simple table, which enables the data to be directly processed by many statistical tools. The GUI for annotators will be released as well.



```
t-root   [ mwes/st/t.rf t-node
 [ parent $n, (a/aux.rf = 'v' or a/aux.rf = 'na'],
 mwes/st/t.rf t-node $n:=
 [ t_lemma = 'stát' ] ];
```

Figure 2: This query searches for all annotated MWEs, such that it consists of the phrase “stát v …” or “stát na …” (meaning “stand in/at/on” as well as “hold ground”, “keep sentinel”, “tiptoe” etc.).



```
t-root $n0:=   [ member mwes/st
 [ lexicon-id ~ 'place',
 mwes/st/t.rf t-node $n:=
 [ t_lemma = 'náměstí' ],
 1+x mwes/st/t.rf t-node $n1:=
 [ ancestor $n ] ] ];
```

Figure 3: A query searching for a NE of a place containing the word “náměstí” (square) and at least one other word.

5 Conclusion

Identification of MWEs moves the Prague Dependency Treebank towards better separation of tectogrammatical lemmas from the morphological lemmas and thus closer to the Functional Generative Description (Sgall et al., 1986), i.e. the theoretical framework the PDT was built upon. In future, this work will help to produce such PDT t-layer, where all units will correspond to whole NEs or lexemes (and so some of them will be multiword).

We employed several different methods to optimise the annotation both in terms of speed and precision. We will continue further refinements of tectogrammatical lemmas before the next release of the treebank.

Acknowledgement

This work has been supported by grant 1ET201120505 of Grant Agency of the Academy of Science of the Czech Republic, and grants GAUK 4307/2009 and GAUK 4200/2009 of the Grant Agency of Charles University in Prague.

References

Eduard Bejček, Pavel Straňák, and Pavel Schlesinger. Annotation of multiword expressions in the prague dependency treebank. In *IJCNLP 2008 Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 793–798, 2008. 1

Eduard Bejček and Pavel Straňák. Annotation of multiword expressions in the prague dependency treebank. *Language Resources and Evaluation*, 43(3), 2009. 2

Jan Hajič. *Insight into Slovak and Czech Corpus Linguistics*, chapter Complex Corpus Annotation: The Prague Dependency Treebank, pages 54–73. Veda Bratislava, Slovakia, 2005. ISBN 80-224-0880-8. 1.1

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA, 2009. 4

Jan Hajič, Jarmila Panevová, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. Prague Dependency Treebank 2.0, 2006. Published by Linguistic Data Consortium, Philadelphia, PA, USA. 1

Milena Hnátková. Značkování frazémů a idiomů v Českém národním korpusu s pomocí Slovníku české frazeologie a idiomatiky. *Slovo a slovesnost*, 2002. 2

Jana Kravalová, Magda Ševčíkov á, and Zdeněk Žabokrtský. Czech named entity corpus 1.0, 2009. 2

Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. Annotation on the tectogrammatical level in the prague dependency treebank. annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep., 2006. 1.1

Petr Pajas and Jan Štěpánek. A Generic XML-Based Format for Structured Linguistic Annotation and Its Application to Prague DependencyTreebank 2.0. Technical Report TR-2005-29, ÚFAL MFF UK, Prague, Czech Rep., 2005. 4

Petr Pajas and Jan Štěpánek. Recent advances in a feature-rich framework for treebank annotation. In Donia Scott and Hans Uszkoreit, editors, *The 22nd International Conference on Computational Linguistics - Proceedings of the Conference*, volume 2, pages 673–680, Manchester, UK, 2008. The Coling 2008 Organizing Committee. ISBN 978-1-905593-45-3. 4

Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. Zpracování pojmenovaných entit v českých textech (treatment of named entities in czech texts). Technical Report TR-2007-36, ÚFAL MFF UK, Prague, Czech Republic, 2007. 2

Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Academia/Reidel Publ. Comp., Praha/Dordrecht, 1986. 5

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly modular MT system with tectogrammatics used as transfer layer. In *ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170, Columbus, OH, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-09-1. 2

Converting a Dependency Treebank to a Categorial Grammar Treebank for Italian

Johan Bos
Bologna, Italy
bos@meaningfactory.com

Cristina Bosco
Università Torino
bosco@di.unito.it

Alessandro Mazzei
Università Torino
mazzei@di.unito.it

Abstract

The Turin University Treebank (TUT) is a treebank with dependency-based annotations of 2,400 Italian sentences. By converting TUT to binary constituency trees, it is possible to produce a treebank of derivations of Combinatory Categorial Grammar (CCG), with an algorithm that traverses a tree in a top-down manner, employing a stack to record argument structure, using Part of Speech tags to determine the lexical categories. This method reaches a coverage of 77%, resulting in a CCGbank for Italian comprising 1,837 sentences, with an average length of 22,9 tokens. The CCGbank for English has proven to be a useful tool for developing efficient wide-coverage parsers for semantic interpretation, and the Italian CCGbank is expected to be an equally useful linguistic resource for training statistical parsers.

1 Introduction

Treebanks have played an important role in the development of robust parsers exploring statistical methods to achieve wide coverage. The key example in this tradition is the Penn Treebank [10], a large collection of English sentences taken from the Wall Street Journal annotated with syntactic structures. The aim of this article is to present the first version of an Italian treebank based on categorial grammar, translated from an existing manually crafted dependency-based treebank [4]. We focus on the various translation steps required to achieve a high-quality treebank.

Our treebank is based on CCG, combinatory categorial grammar [12], a lexicalised grammar formalism encoding all non-local dependencies in its lexical categories. A further motivation for using CCG is its transparency between syntactic categories and semantic types, providing an ideal platform for automatically building formal semantic representations [2]. Other treebanks based on categorial grammar have been developed in the past, of which the English CCGbank [7] derived from the Penn Treebank [10] is the prime example. The English CCGbank has proven to be a useful resource for training robust parsers [6], and we follow its design as closely as possible.

For automatically deriving Italian categorial grammar, previous work on has been carried out by [1], who provide a method for translating TUT dependency structure to derivations of type-logical grammar, but only for a relatively small set of examples with low structural complexity, producing a lexicon of 1,909 words based on 400 derivations with an average of two categories per words. Even though our work is similar in spirit, we also deal with more complex cases extending the coverage considerably.

2 Background

2.1 The Turin University Treebank (TUT)

The starting point of our conversion in CCG is TUT, the Turin University Treebank[1]. This Italian treebank currently includes 2,400 sentences corresponding to around 72,150 tokens. This annotated corpus consists of three parts: 1,100 sentences from newspaper texts (mainly from *La Stampa* and *La Repubblica*), 1,100 sentences from the Italian Civil Law Code, and 200 sentences from the Italian section of the JRC-Acquis corpus.[2]

The development of TUT has its roots in a dependency-based annotation following the major principles of Word Grammar [8]. Central to this is a notion of argument structure described by a rich set of grammatical relations that can include three components: morpho-syntactic, functional-syntactic, and semantic information [3]. The TUT annotation process includes automatic POS tagging and parsing [9], completed with a set of automatic and manual correctness and consistency checks.

To promote applications of TUT, several efforts have been directed to automatically converting the treebank into other formats. Among these are bracketed labelling known from the Penn Treebank [10], as well as the Xbar-like format called Constituency TUT (henceforth ConsTUT), which forms an important step in converting TUT to CCG. These mappings to other formats have not only increased the comparability with other existing linguistic resources, but also improved the quality of the annotated material. Even though the size of TUT is relatively small compared to the Penn Treebank, the EVALITA 2009 shared task on parsing showed that dependency parsers trained on TUT are close to the state-of-the-art [5].

2.2 Combinatory Categorial Grammar (CCG)

CCG is a lexicalised theory of grammar in which all syntactic dependencies are encoded in the lexical categories [12]. The version of CCG that we adopt is based on the English CCGbank [7], comprising a set of CCG derivations derived from

[1]See `http://www.di.unito.it/~tutreeb/`, [4].

[2]See `http://langtech.jrc.it/JRC-Acquis.html`.

the Wall Street Journal texts from the Penn Treebank [10]. The basic categories are `s` (sentence), `n` (noun), `pp` (prepositional phrase), `np` (noun phrase) and `t` (text). Functor categories are composed out of the basic categories with the help of slashes indicating order and position of arguments: a functor category $\alpha\backslash\beta$ yields a category α when it finds an argument of category β on its left, and a functor category α/β yields a category α when it finds an argument of category β on its right. We follow the convention introduced in CCGbank and associate the category `s` with a feature indicating sentence mood or aspect of verb phrases (Fig. 1).

```
                                                                tutta       balcanica
                                                                ------[lex]  ----[lex]
                                                       aria     (n\n)/(n\n)  n\n
                                                       --[lex]  ------------------[fa]
                  questa   vicenda            un’      n        n\n
                  --[lex]  --[lex]            --[lex]  ---------------------------[ba]
In                np/n     n        tira      np/n     n
-----------[lex]  ------------[fa]  ---[lex]  ------------------------------------[fa]
(s:dcl/s:dcl)/np  np                s:dcl/np  np
------------------------------[fa]  ----------------------------------------------[fa]
s:dcl/s:dcl                         s:dcl                                              .
----------------------------------------------------------------------------------[fa] --[lex]
s:dcl                                                                                  t\s:dcl
------------------------------------------------------------------------------------------[ba]
t
```

Figure 1: CCG derivation for *In questa vicenda tira un’ aria tutta balcanica*.

To combine categories deriving new categories, CCG is equipped with a small set of combinatory rules and a couple of non-combinatory rules. The combinatory rules combine two categories and produce a new one. They comprise forward application (>), backward application (<), forward and backward composition (>B and <B), forward and backward substitution (>S and <S), their crossing variants, and generalised versions of the composition rules. All of these rules have a direct semantic interpretation, and give expressive power that go beyond context free grammars [12].

The non-combinatory rules consist of the type-raising and type-changing rules. They are unary rules, mapping a single category into a new one. The type-raising rules (>T and <T) change an argument in a functor in a systematic way. The type-changing rules are used for covering elliptical expressions, such as pro-drop, adjective-like participles, and determinerless noun phrases.

3 Method

We take as input a set of sentences in the TUT dependency format. These are first mapped onto constituency trees, then transformed into binary trees, and finally converted into CCG derivations.

3.1 From Dependency Structures to Constituency Trees

ConsTUT is a constituency-based annotation with constituents decorated by TUT relations. In ConsTUT trees each terminal category X corresponds to a node (i.e.

token) of a TUT tree, and projects into non-terminal nodes which represent intermediate (Xbar) and maximal (XP) projections of X, following Xbar theory [11]. Each node is classified as either head (h), argument (a), or modifier (m). To illustrate this projection of categories, consider for instance the adverb *tutta* in Fig. 2. This adverb projects on ADVbar and then on ADVP, as the ConsTUT tree in Fig. 3 shows.

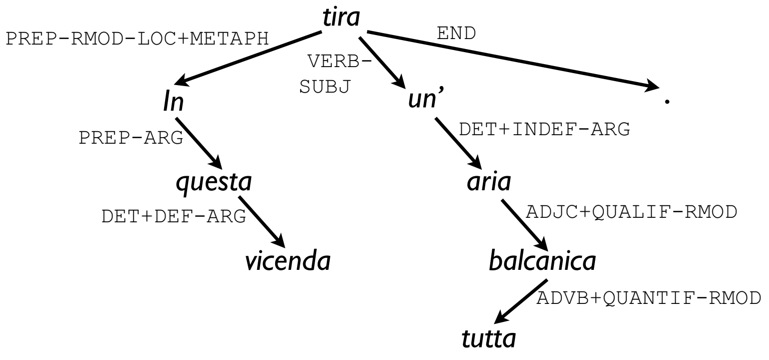


Figure 2: TUT A-6, *In questa vicenda tira un' aria tutta balcanica*.

Further following Xbar theory, the distinction between arguments and modifiers is structurally marked. Arguments, usually closer to their head, are daughters of intermediate projections and sisters of terminal categories. Modifiers, on the other hand, are both sisters and daughters of intermediate projections. Schematically, this can be viewed as: [XP (Xbar (Xbar (X)(ARG)) (MOD))].

The conversion algorithm from TUT to ConsTUT is adapted from [13], who employed it for the conversion of dependency structures in Penn Treebank style phrase structures, i.e., constituents featuring a minimal projection strategy. The input of our algorithm are ordered dependency trees, that is projective structures where dependents of the same head are ordered according to their positions in the sentence. Explicit marking of null elements can occur to resolve cases of non-projectivity too. The output of the algorithm are constituency trees applying a maximal projection strategy.

The main information that is present in a constituency tree, but not in a dependency tree, is the type of non-terminal nodes (e.g. NP, VP and S). Therefore, the major goal of the algorithm is to recover the types of phrases that each node of the dependency tree projects. In other words, the task here is to find the expansions in constituency terms of the grammatical category of terminal nodes, which have to be annotated as non-terminal nodes in the constituency trees. The trees also contain Part of Speech (POS) tags on terminal nodes, using a simplified tagset of that used in TUT [3].

Grammatical categories can interact in dependency trees. To build a corresponding constituency tree, one needs to know how one can represent this inter-

```
h(S
   a(DP-SUBJ*[533])
   h(Sbar
       h(VP
           h(Vbar
              m(PP-RMOD-LOC+METAPH
                  h(PREP 1/In)
                  a(DP-ARG
                     h(Dbar
                        h(ADJ~DE 2/questa)
                        a(NP-ARG
                            h(Nbar
                                h(NOU~CS 3/vicenda))))))
              h(Vbar
                  h(VMA~RE 4/tira)
                  a(DP-EXTRAPOSEDSUBJ*533
                      h(Dbar
                          h(ART~IN 5/un')
                          a(NP-ARG
                              h(Nbar
                                  h(Nbar h(NOU~CS 6/aria))
                                  m(ADJP-RMOD
                                     m(ADVP-RMOD
                                        h(ADVbar h(ADVB 7/tutta)))
                                     h(ADJbar
                                        h(ADJ~QU 8/balcanica))))))))))
   m(PUNCT-END 9/.))
```

Figure 3: ConsTUT tree for *In questa vicenda tira un' aria tutta balcanica*.

action in constituency terms, that is, how grammatical categories and their projections combine in constituency structure. This is governed by language specific constraints, depending on the types of modifiers and arguments that a head can take and their positions related to the head itself. All this information is encoded in a look-up table that the converter exploits.

3.2 From Constituency Trees to Binary Trees

The lexical categorisation algorithm, which is last in the pipeline, requires binary trees (trees with at most two branches) as input. The syntactic analyses of ConsTUT are represented by n-ary trees, but not necessarily binary trees (see Fig. 3). Given the distinction between head, modifier and argument, there are four possible kinds of trees (Fig. 4).

```
     .                .                .                .
    / \              / \              / \              / \
  H/   \M          M/   \H          H/   \A          A/   \H
  /     \          /     \          /     \          /     \
 .       .        .       .        .       .        .       .
```

Figure 4: Possible binary sub-trees.

To map a tree with n ($n > 2$) branches into a tree with $n - 1$ branches, we select a HEAD-X or X-HEAD sequence from the ordered list of branches (where X is ARG or MOD), and replace it by a new branch marked by HEAD forming a binary tree of the two selected branches. This process is iterated for each node until it is completely binary. The procedure is illustrated for a ternary tree mapped to a binary tree in Figure 5.

```
                                   .
      .                          H/ \
     /|\                         .    \
    / | \          =>           / \    \
 H/   |X \                    H/   \X   \
 /    |   \                   /      \   \
.     .    .                 .       .    .
```

Figure 5: Producing binary trees.

This is a general rule — there are specific rules that cover awkward cases such as punctuation. At this stage we also deal with Italian "definite prepositions" such as *sulla*, which are contractions of a preposition and a definite article. In ConsTUT these appear as two different nodes (PREP and ART-DE) in the tree, related to each other by co-indexing. These two nodes are mapped onto a newly marked node (DEFPREP) in order to distinguish it from ordinary prepositions in the categorisation process.

3.3 From Binary Trees to Categorial Grammar Derivations

The categorisation algorithm takes a binary constituency tree as input and produces a CCG derivation. The core of the algorithm is based on "pure" categorial grammar, using only forward application (>) and backward application (<) of the set of combinatory rules. In a nutshell, the algorithm proceeds as follows: it traverses the binary input tree in a top-down fashion, starting with the root node. The final values for the categories for the nodes, however, are generated via a bottom-up strategy, determined by the POS tags. The algorithm makes use of a stack on which it pushes the categories of arguments encountered in the binary tree. The elements of the stack determine the lexical categories of heads.

There are four general rules that deal with the standard cases in Fig. 4, which can be overridden by more specific rules to cover special linguistic constructions, such as punctuation or coordination. First consider the two modifier cases, where the algorithm produces a CCG derivation for the head first. Suppose this yields category X, then we make the category for the modifier X\X in the HEAD-MOD case, introducing the backward application rule (<). The MOD-HEAD case forms a mirror image of this mapping, yielding the category X/X for the modifier by virtue of the forward application rule (>). Note that, in both cases, nothing is altered to the value of the stack (S); it is just passed on when translating HEAD (Fig. 6).

In the HEAD-ARG and ARG-HEAD cases, the argument is analysed first. The resulting category is pushed on the stack (plus its direction: \ or /). Suppose that the translation of ARG yields the category Y, then the HEAD-ARG case maps to a

```
          .                  ccg(T1,S)      ccg(T2,[])
         / \                 ----------     -----------
ccg(   H/    \M    ,S)   ⇒      X             X\X
       /      \              -------------------------<
     T1        T2                        X
```

```
          .                  ccg(T1,[])       ccg(T1,S)
         / \                 -----------      ----------
ccg(   M/    \H    ,S)   ⇒     X/X               X
       /      \              -------------------------->
     T1        T2                     X
```

Figure 6: The modifier case for translating binary trees to CCG.

forward application rule with X/Y being the category for the head constituent. The ARG-HEAD case is a mirror image of the HEAD-ARG translation rule. Here we introduce the backward application rule with X\Y being the category for HEAD, and the category produced for ARG is pushed onto the stack. This is illustrated by Fig. 7.

```
         .             ccg(T1,push(/Y,S))    ccg(T2,[])
        / \            -------------------   -----------
ccg(  H/    \A   ,S) ⇒      X/Y                   Y
      /      \         ---------------------------------->
    T1        T2                     X
```

```
          .            ccg(T2,[])     ccg(T1,push(\Y,S))
         / \           -----------    -------------------
ccg(   A/    \H  ,S) ⇒     Y                X\Y
       /      \        ----------------------------------<
     T1        T2                 X
```

Figure 7: The argument case for translating binary trees to CCG.

These four general rules are used in traversing the binary tree in a top-down fashion. Once a leaf node is reached, the lexical categories are determined, which in turn provide information to determine the values of the non-lexical categories encountered earlier. The mapping from a leaf node to a lexical CCG category depends on whether it is playing the role of head, argument, or modifier. A leaf node of type MOD is either of the form X/X or X\X, where X is already determined by the head using either the HEAD-MOD or MOD-HEAD rule. In the case of ARG, the lexical category is determined by the assigned Part of Speech in TUT (see Table 2). For HEAD we follow the same strategy as for ARG, but we resort to the stack to determine the number and direction of arguments. The features introduced on the category `s` are shown in Table 1.

Passive sentences are marked in ConsTUT by empty argument nodes marked as logical subject of a verb phrase. These are pushed onto the stack as were they ordinary arguments. However, when generating a lexical category for a verb phrase, the feature chosen for `s` will be `pss` when a logical subject is a member of the stack. Coordination is dealt with by giving conjunctors the category (X\X)/X, where X can be any category.

Finally, a post-processing procedure deals with empty nodes, clitics, normalisation of accents, and certain kinds of punctuation. Empty nodes, such as Italian pro-drop, introduce type-changing rules in the CCG derivation. Verbal clitics are explicitly marked as arguments combined with an application rule. Opening and closing parentheses and quotes are given categories (X/p)/X and X\(X/p), respectively, where X can be any category. We also normalize the accents post-processing step, since TUT allows for distinct ways of encoding Italian accents.

4 Results

Recall that the conversion process from TUT dependencies to CCG derivations forms a pipeline of three components. These conversion steps are all performed automatically, and hence prone to errors. Reasons for failure in the conversion are complex cases of coordination, clitics, verbal structures, or elliptical phrases. In part these can and will be dealt with in future, revised versions of the treebank, by adding more specific transformation rules to the algorithm.

In Table 3 we report coverage of the conversion process by the number of well-formed trees that we obtain as output after each processing step. The civil law corpus yields the highest coverage, probably because it is a part of TUT containing many relatively short sentences, with less complex, legal language. This claim is confirmed by looking at the length of the sentences in the three corpora: the newspaper corpus has an average length of 24.04 (19,355 tokens on 807 sentences), the civil law corpus 20.97 (18,766 tokens on 896 sentences), and the JRC-Acquis cor-

Table 1: Features on category `s` (verb phrases and sentences).

Part of Speech	**Lexical Category**	**Description**
VMA-GE	`s:ger`	gerund
VMA-PA	`s:pap`	past participle
VMA-PE	`s:prp`	present participle
VMA-RE	`s:dcl`	present
VMA-IN	`s:inf`	infinitive
VMA-IP	`s:imp`	imperative
	`s:adj`	adjective phrase
	`s:pss`	passive
	`s:ynq`	yes/no-question
	`s:whq`	wh-question

Table 2: Mapping from POS tags to lexical categories.

Part of Speech	Lexical Category	Description
VMA	`s`	main verb
VMO	`s`	modal verb
VAU	`s`	auxiliary verb
ART	`np`	article
PRDT	`np`	pre-determiner
PRO	`np`	pronoun
PRO-PO	`np`, `n`	possessive pronoun
NOU	`n`, `np`	common noun
ADJ	`s:adj\np`, `n/n`, `n\n`	adjective
PREP	`pp`, X/X, X\X	preposition
NUMR	`n`, `np`	numeral
PUNCT-END	`t`	punctuation
CONJ	X	conjunction

pus 29.08 (3,839 tokens on 132 sentences). Hence, it is likely that the JRC-Acquis corpus produces a high number of mistakes because the sentences are significantly longer. Moreover, this corpus is only recently been added to TUT, and as a consequence still contains a number of syntactic constructions that are not yet covered by the dependency to constituency conversion step.

Table 3: Number of trees in the three corpora after each processing step.

Corpus	TUT	→	ConsTUT	→	Bin	→	CCG	
Newspaper	1,100	→	890	→	827	→	807	(73%)
Civil Law	1,100	→	993	→	909	→	898	(82%)
JRC-Acquis	200	→	147	→	133	→	132	(66%)

The total number of different lexical categories generated for all three corpora is 1,152, of which 627 occur more than once. For comparison, the English CCG-bank hosts 1,286 different lexical categories, of which 847 occur at least twice [7]. The ten most frequent categories are shown in Table 4, and the average number of categories per token is 1.66.

Since one of the aims of the Italian CCGbank is to produce statistical language models, an important question to ask is whether the size of the treebank is large enough for this purpose. The size of TUT is relatively small compared to the size of the Penn Treebank, but it would be helpful to estimate the required size of a CCGbank as developing annotated tree structures is a costly business. We can't completely answer this question, but what we can do, is look at the number of different categories produced for each corpus to get a rough idea.

We do this in Table 5, where we show the number of different categories, as well as the number of different POS-category pairs. We also computed these values

Table 4: Frequency of the ten most common categories.

Category	Newspaper	Civil Law	JRC-Acquis	Total
n	4,477	4,572	904	9,953
np/n	1,807	1,658	345	3,810
(n\n)/n	1,228	1,050	288	2,566
n\n	1,254	932	365	2,551
pp/n	707	945	255	1,907
t\s:dcl	720	593	122	1,435
np	604	645	50	1,299
n/n	706	377	128	1,211
n/pp	279	359	142	780
s:dcl/s:dcl	297	248	52	597

with (+) and without (−) features on these categories. As the table shows, the numbers differ substantially, clearly caused by the size of the respective corpora. Next, we investigated the relation between the number of lexical categories and the size of the treebank. A treebank with a good coverage would show an emerging plateau in the number of categories by an increase of the number of sentences considered.

Table 5: Number of POS and categories of the three corpora.

	Newspaper	Civil Law	JRC-Acquis	Total
CAT (−)	541	411	188	714
CAT (+)	841	657	291	1,154
POS+CAT (−)	826	640	277	1,121
POS+CAT (+)	1,121	876	362	1,576

As Fig. 8 shows, the growth of categories is clearly decreasing, yet rising. This is caused in part by the small size of the treebank, but we think it is also due to the absence of forward and backward (crossed) composition rules in the categorisation algorithm implemented so far. Adding such combinatory rules for specific but common linguistic structures is part of future work, as is abstracting over modifier and conjunction categories, to gain a further reduction of distinct lexical categories [7]. The current version of the Italian CCGbank is released under a creative commons license via `http://www.di.unito.it/~tutreeb/CCG-TUT/` in various formats: the derivations printed in human-readable format (as in Fig. 1), derivations printed as Prolog terms, or as token-POS-category tuples.

Acknowledgements We thank Jason Baldridge, James Curran, Leonardo Lesmo, Malvina Nissim, and Mark Steedman for their comments on earlier versions of this paper and their feedback on the Italian CCGbank.

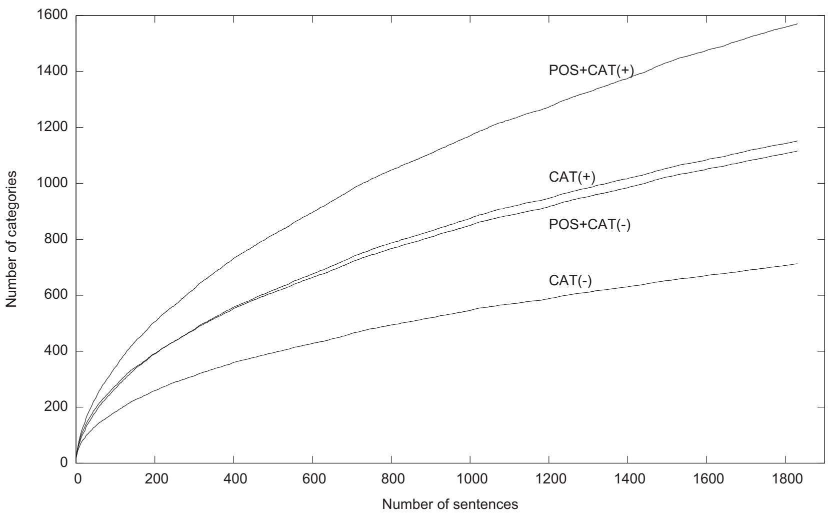


Figure 8: Growth of lexical categories with respect to number of sentences.

References

[1] R. Bernardi, A. Bolognesi, C. Seidenari, and F. Tamburini. Learning an italian categorial grammar. In R. Rossini Favretti, editor, *Frames, Corpora, and Knowledge Representation*, pages 185–200. 2008.

[2] J. Bos. Wide-Coverage Semantic Analysis with Boxer. In J. Bos and R. Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications, 2008.

[3] C. Bosco. *A grammatical relation system for treebank annotation*. PhD thesis, University of Torino, 2004.

[4] C. Bosco, V. Lombardo, D. Vassallo, and L. Lesmo. Building a Treebank for Italian: a Data-driven Annotation Schema. In *Proc. 2nd Int. Conf. on Language Resources and Evaluation*, pages 99–105, Athens, 2000.

[5] C. Bosco, S. Montemagni, A. Mazzei, V. Lombardo, F. Dell'Orletta, and A. Lenci. Evalita'09 parsing task: comparing dependency parsers and treebanks. In *Proceedings of EVALITA 2009*, Reggio Emilia, 2009.

[6] S. Clark and J.R. Curran. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain, 2004.

[7] J. Hockenmaier. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, Univ. of Edinburgh, 2003.

[8] R. Hudson. *Word Grammar*. Blackwell, Oxford and New York, 1984.

[9] L. Lesmo. The rule-based parser of the NLP group of the university of Torino. *Intelligenza Artificiale*, IV(2):46–47, 2007.

[10] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[11] A. Radford. *Syntactic theory and the structure of English. A minimalist approach*. Cambridge University Press, 1997.

[12] M. Steedman. *The Syntactic Process*. The MIT Press, 2001.

[13] F. Xia. *Automatic grammar generation from two different perspectives*. PhD thesis, University of Pennsylvania, 2001.

Dynamic Propbanking with Deep Linguistic Grammars

António Branco, Sara Silveira, Sérgio Castro, Mariana Avelãs, Clara Pinto and Francisco Costa

University of Lisbon
`http://nlx.di.fc.ul.pt`

Abstract

In the development of annotated corpora with deep linguistic representations, the category to be assigned to each markable (i.e. the fully fledged grammatical representation of each sentence) is so complex that it cannot be safely constructed manually and the annotation cannot be performed without some supporting application, viz. a computational grammar. This paper discusses and present solutions for the adaptation of deep linguistic grammars as supporting tools in the construction of dynamic propbanks. It reports on the results of an experimental study with a pilot application of this approach to propbanking. Estimated scores for inter-annotator agreement and for development effort are presented.

1 Introduction

Stochastic parsers rely on the availability of annotated corpora both for their training and their evaluation. Such corpora may encompass linguistic information of varying level of complexity, ranging from relatively shallow representation of syntactic constituency (e.g. Penn treebank [12]) to deep representation of fully fledged grammatical analysis that includes advanced semantic representation (e.g. Redwoods HPSG treebank [14]).

The expected trend is that the corpora supporting grammatical analysis will be language resources bearing an increasingly sophisticated linguistic information (Treebanks, PropBanks, LogicalFormBanks, ...). This is in line with the trend observed in natural language processing in general, where annotated corpora that have been developed address issues well beyond the initial part-of-speech level, including issues from the realm of semantics (e.g. corpora annotated with coreference links [11]) and from the realm of discourse (e.g. corpora annotated with dialogue acts tags [7]).

To develop corpora that are annotated with deep linguistic representations, the fully fledged grammatical representation to be assigned to each sentence is so complex and so specific that it cannot be reliably crafted manually piece by piece and the annotation cannot be performed without some supporting application. Such annotation environment has to integrate at least a computational grammar to construct representations that cannot be done by hand, and a user interface to display the parses available and to allow the annotator to selected one of them (e.g. the `[incr tsdb()]` application [13]).

Propbanks are treebanks whose trees have their constituents labeled with semantic role tags. In other words, propbanks are annotated corpora that result from the extension of the annotation associated to the sentences in treebanks by means of an extra set of tags for semantic roles. The seminal work in this area is due to [15] and the semantic roles adopted for arguments were ARG1, . . . , ARG4, while the semantic roles used to classify modifiers include tags such as LOC, TMP, MNR, etc.

The construction of propbanks thus offers a new area for the application of deep linguistic grammars to the development of language resources. It also presents specific challenges that need to be addressed.

While word sense disambiguation is a task of semantic categorization for words, propbanking can be seen as a task of semantic categorization for phrases. At their core, tasks of semantic categorization involve semantic ambiguity resolution and deep linguistic grammars are known not to be the most suitable approach to handle semantic categorization tasks. Moreover, any deep linguistic grammar that might be used to support (at least the initial stages of) the construction of a propbank could not rely on stochastic models that would allow it to resolve semantic role assignment, as the training data needed to obtain these models are the materials that are being constructed with the help of the grammar.

An immediate move to ponder would be to extend the grammar so that the different parse trees it produces for a given sentence have their constituents labeled with the different possible semantic roles. The task of propbanking would then be similar to the task of treebanking. The human annotator would go through the parse forest generated and select the appropriate tree, with the appropriate semantic roles in the appropriate constituents. However, this turns out not to be a practically viable option as it would induce an exponential explosion of the size of the parse forest and would severely hurt the efficiency of the grammar.

An alternative to this one-step approach is a two-phase approach. In a first phase, the grammar is used to get at the correct tree for the sentence at stake. Once that representation is selected, in a second phase, its phrases are tagged by the human annotator. This is a scenario that avoids the explosion of the parse forest that would result from the one-step approach. And very importantly, this is a scenario that still allows exploring the contribution of the grammar for propbanking to the maximum extent possible without spurious overgeneration.

The grammar can be used not only to construct the underlying syntactic tree but also to advance the assignment of semantic roles: if its deep linguistic representation is "deep" enough to include the predicate-argument structure of the sentence, the grammar can be used to correctly assign the role labels for its constituents that are arguments, viz. ARG1 to ARG*n*. By the same token, it can be used to identify the phrases that are modifiers, and hence to automatically select only those constituents that still need to be further manually inspected and specified for their semantic role.

In this paper we report on the adoption of this approach to propbanking and describe the solutions we developed in order to construct a propbank based on a deep linguistic grammar. This involved using an existing deep linguistic grammar and using available tools for the creation of dynamic treebanks, that accommodate the fact that the grammar may evolve and its output may be altered and refined.[1] Accordingly, in this paper we will also describe the solutions developed for the construction of a dynamic propbank that is supported by a grammar that may evolve and be refined along time.

Section 2 reports on the adaptation of a deep linguistic grammar to the task of semi-automatic propbanking and of the tool for dynamic treebanking. In Section 3, we describe the annotation environment developed to assist in the manual completion of a dynamic propbank. Section 4 describes a pilot experimental study of the practical viability of this environment, and Section 5 presents the concluding remarks.

2 Semantic role tagging by the grammar

The deep linguistic grammar used for semi-automatic propbanking was LXGram [5, 6, 4, 3]. This grammar is developed under the grammatical framework of HPSG-Head-Driven Phrase Structure Grammar [16, 17] and uses MRS-Minimal Recursion Semantics [10] for the representation of meaning. Its implementation is undertaken with the grammar development environment LKB-Linguistic Knowledge Builder [9] and the LinGO Grammar Matrix version 0.9 [2] was used as the initial code upon which the grammar is being built.

The evaluation and regression testing of this grammar is done via the application `[incr tsdb()]` [13] that works in tandem with the LKB. Once the sentences are parsed, they are manually disambiguated using this profiling environment, which can then be used to export into text files several views of the syntactic and the semantic analyses obtained by the grammar (e.g. parse trees, feature structures and semantic representations).

As this is an HPSG grammar, there are no explicit categories like Sentence, Noun Phrase, etc. in the grammatical representation. Instead there are feature structures that describe and stand for such categories. The LKB environment, however, has a visualization device for grammatical representations that permits to display tree views where these categories can be made to appear. There are specific configuration files for this device where it can be stated what feature structures should be mapped to what symbols that appear in the nodes of the syntax

[1]We are using the notion of "dynamic" annotation of corpora (for treebanking, propbanking, etc) in the sense worked out for the Redwoods Treebank in [14].

tree displayed. For instance, a constituent with a HEAD feature of the type *verb* and a SUBJ feature with the empty list as its value is categorially a sentence, and it is possible to configure the visualization device for the corresponding tree node to appear with the label S. This is the key facility we explored in this grammar development environment to make semantic role labels appear in the appropriate constituents in the parse tree.

Some of the semantic role labels that are used in PropBank can be obtained from features that describe the semantics of the sentence, namely those used to tag the subject and the complements of predicators, ARG1 to ARGn.

For instance, a verb that is associated to a semantic relation whose first argument is that verb's subject will comply with a constraint like the following:

$$\left[\text{SYNSEM}|\text{LOC}\ \begin{bmatrix}\text{CAT}|\text{VAL}|\text{SUBJ}\left\langle\left[\text{LOC}|\text{CONT}|\text{HOOK}|\text{INDEX}\ \ \boxed{1}\right]\right\rangle\\ \text{CONT}|\text{RELS}\left\{\left[\text{ARG1}\ \ \boxed{1}\right]\right\}\end{bmatrix}\right]$$

This is the underlying piece of information that can be used to assign semantic role labels. Although this piece of information is visible in the feature structures for predicators, it is not visible in the feature structures for the actual phrases that are to be labeled. For this reason, the grammar was expanded with an extra feature (ROLE-LABEL) marking the semantic role label of constituents. In this example, with this sort of verbs, another constraint was added:

$$\left[\text{SYNSEM}|\text{LOC}\ \begin{bmatrix}\text{CAT}|\text{VAL}|\text{SUBJ}\left\langle\left[\text{LOC}\ \begin{bmatrix}\text{ROLE-LABEL} & \textit{arg1}\\ \text{CONT}|\text{HOOK}|\text{INDEX} & \boxed{1}\end{bmatrix}\right]\right\rangle\\ \text{CONT}|\text{RELS}\left\{\left[\text{ARG1}\ \ \boxed{1}\right]\right\}\end{bmatrix}\right]$$

A quite straightforward way to include such semantic role labels in the output tree is by simply adjusting the mapping from feature structures to node labels in the visualization device so that the information contained in the semantic representations is rendered visible in the tree exported.

For the sake of illustration, Figure 1 presents an example of a parse tree exported. All constituents have a category label. The relevant constituents also have a tag describing their syntactic function (SJ for subject, DO for direct object, IO for indirect objects, OBL for oblique complements, PRD for predicative, M for modifier), separated from the category label by a dash. Also separated by a dash, a third tag describes the semantic role that can be obtained from the semantic representations derived by the grammar, ARG1,. . . , ARG*n* for subcategorized arguments, M for modifiers.

The ARG1,. . . , ARG*n* tags are similar to the same tags used in PropBank, but note that PropBank starts at ARG0 whereas we start at ARG1. The M is here a portmanteau tag that covers all semantic roles that are possible for modifiers. It corresponds to PropBank tags like LOC, TMP, MNR, etc. As mentioned above, producing these more specific tags by the grammar is not practically viable. Our approach to propbanking consists in manually refining the M tag in a second phase, that is described in the next section.

```
(S (PP-M-M (P (Para))
           (S (NP-SJ-ARG1 (D-SP (a))
                          (N (delegação)))
              (VP-C (V (evitar))
                    (NP-DO-ARG2
                      (D-SP (um))
                      (N′ (N (conflito))
                          (AP-M-M (A (armado))))))))
   (S (PP-M-M (P (em))
              (NP-C (N (Maio))))
      (S (NP-SJ-ARG1 (D-SP (a))
                     (N (ONU)))
         (VP (V′ (V′ (V (enviou))
                     (ADVP-M-M (ADV (rapidamente))))
                 (NP-DO-ARG2 (N (tropas))))
             (PP-OBL-ARG3
               (P (para))
               (NP-C (D-SP (a))
                     (N (fronteira))))))))
```

Figure 1: Parse tree exported from `[incr tsdb()]` and decorated with semantic role labels. The sentence translates into "For the delegation to_avoid an armed conflict, in May the UN quickly sent troops to the border".

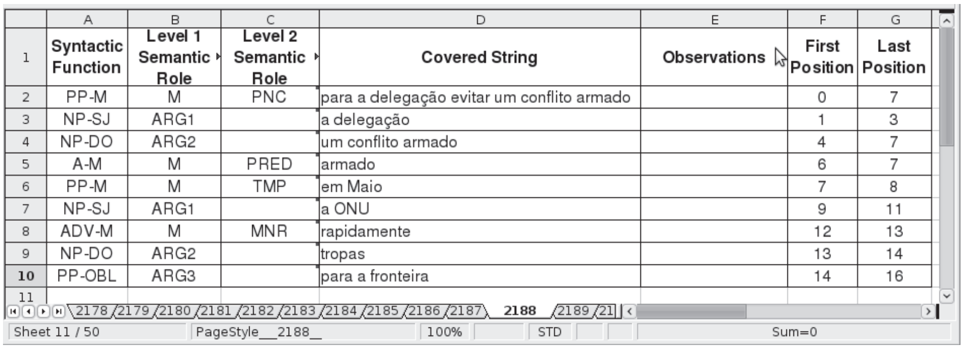
3 Completing semantic role tagging

After the propbanking is advanced in a first phase by the grammar, a completion step follows that consists in the manual specification of the occurrences of the portmanteau tag M in terms of one of the semantic roles available for modifiers in the tagset, LOC, TMP, MNR, etc.

Before proceeding with the descritpion of the second phase of the propbanking, it is worth noting that for the two step annotation methodology we are reporting on, there is nothing essential in the usage of the HPSG framework, the LKB development environment or an MRS semantic representation. Any deep linguistic grammar is suitable to support the first, automatic phase described in the section above provided it delivers deep enough representations, that include at least the semantic roles for arguments.

Turning now to the second phase of the propbanking, this manual annotation is prepared by two tools. A converter from trees into an annotation format compatible with the annotation interface, and a reverser tool for the symmetric operation.

A third tool is also necessary to support the construction of a dynamic Propbank. The development of this annotated corpus is based on a computational grammar which is itself likely to be under development or refinement and to evolve. It should thus be expected that a sentence that received a certain analysis under version n of the grammar may receive a different tree under the subsequent version n+1. This change in grammatical analysis is likely to have an impact on the annotation produced with the support of the grammar in version n and previously stored in the Propbank. Therefore, a third tool is necessary to support this dynamic



	A	B	C	D	E	F	G
1	Syntactic Function	Level 1 Semantic Role	Level 2 Semantic Role	Covered String	Observations	First Position	Last Position
2	PP-M	M	PNC	para a delegação evitar um conflito armado		0	7
3	NP-SJ	ARG1		a delegação		1	3
4	NP-DO	ARG2		um conflito armado		4	7
5	A-M	M	PRED	armado		6	7
6	PP-M	M	TMP	em Maio		7	8
7	NP-SJ	ARG1		a ONU		9	11
8	ADV-M	M	MNR	rapidamente		12	13
9	NP-DO	ARG2		tropas		13	14
10	PP-OBL	ARG3		para a fronteira		14	16
11							

Figure 2: Annotation interface

propbanking.

These tools are described in the next subsections.

3.1 Exporting trees to and from an annotation interface

The annotation interface is based on a basic yet very efficient and powerful enough technology in view of the manual task it is aimed at supporting. A set of sentences to be annotated is presented in a spreadsheet file, with each sentence in a different sheet. The constituents that need to be tagged are displayed in the first column, each constituent in a separate cell. The semantic role labels that were assigned by the grammar are displayed in the second column, aligned with the corresponding constituents. The third column is left blank and its cells in the lines with M in the second column offer a drop down menu from which it is possible to pick the semantic role label with which to specify M.

These spreadsheets are created by the converter tool that takes as input an exported version of the sentences treebanked with `[incr tsdb()]`. For each suite of treebanked sentences, a spreadsheet is created with as many sheets as sentences in that suite. If a given sentence happens not to have received a parse, its sheet only contains its identification number and that sentence. If in turn the sentence received a parse in the treebank, its tree is processed and for each node with a syntactic function that ends in -SJ, -DO, -IO, -M, -OBL, or -PRD, a new line in the sheet is printed.

The sentences that have a parse, i.e. whose propbanking needs to be finalized, are identified by a designated format for the name of the sheets containing them. The annotator only needs to look for those sheets with the name in that format, since only there actual annotation is required to be performed.

As mentioned above, each line has cells automatically filled in, and others to be filled in by the annotator. Each line includes cells with (A) the syntactic category and grammatical function, (B) the semantic role assigned by the grammar, (C) the “level two” of the semantic annotation—the cell to be filled in by the human annotator—, (D) the constituent being tagged, (E) the annotator’s observations—

```
(S (PP-M-PNC (P (Para))
             (S (NP-SJ-ARG1 (D-SP (a))
                            (N (delegação)))
                (VP-C (V (evitar))
                      (NP-DO-ARG2
                        (D-SP (um))
                        (N′ (N (conflito))
                            (AP-M-PRED (A (armado)))))))
   (S (PP-M-TMP (P (em))
                (NP-C (N (Maio))))
      (S (NP-SJ-ARG1 (D-SP (a))
                     (N (ONU)))
         (VP (V′ (V′ (V (enviou))
                     (ADVP-M-MNR (ADV (rapidamente))))
                 (NP-DO-ARG2 (N (tropas))))
             (PP-OBL-ARG3
               (P (para))
               (NP-C (D-SP (a))
                     (N (fronteira))))))))
```

Figure 3: Parse tree after manual role labeling. Newly introduced tags are -PNC, -PRED, -TMP and -MNR.

free text by the annotator—, and finally (F-G) the begin and the end positions of that phrase in the sentence.

When the propbanking is finalized, the sentences are reverted to the original tree representation, now extended with the newly assigned tags for the semantic roles of modifiers.

This operation is ensured by a reverting tool. This tool parses the data in the sheets of the spreadsheet and, guided by the information in the last column of the sheets on the initial and final position of the phrase, recombines the information stored there with the original information about the parse tree of the sentence. The outcome is a set of restored sentence trees like the one displayed in Figure 1 with the only difference that now level 2 tag M does not occur in them, as in Figure 3. Each one of its occurrences in the original tree generated by the grammar was replaced by the tag with which it was manually specified.

Note that this annotation interface is very flexible and permits to accommodate tagsets other than the one adopted here, which replicates the tagset of the English PropBank in [15]. It permits to specify not only the modifiers with sub-semantic roles LOC, TMP, MNR, etc., but also the arguments with sub-semantic roles like AGENT, PATIENT, EXPERIENCER, etc. One just needs to add extra drop down menus with the required range of tags in the third column cells of the lines already containing the ARG1,..., ARGn tags.

This is thus an annotation interface that can very easily be adapted for arbitrarily complex, multi-layer hierarchical tagsets of semantic role labels, including those that may be extended and get more complex along the development of refined versions of the propbank.

3.2 Supporting the construction of a dynamic propbank

An annotated corpus whose construction is based on a grammar that evolves with time has a dynamic nature. As the grammar gets extended or refined in each new version, the composition of the corpus is likely to evolve as well. For instance, some sentences that got a parse with a previous version n and were annotated with that parse may have no parse in version n+1; sentences that received a parse with version n of the grammar may have that parse tree altered in version n+1; and sentences that had no parse with version n may receive a parse in version n+1. As the grammar evolves, sentences may thus be dropped from the annotated corpus, may have their annotation changed, or new sentences may enter the annotated corpus. Of course, many annotated sentences in the treebank will be kept unchanged in the new versions.

Given the annotation environment adopted for propbanking, in order to minimize the manual effort spent, the two cases that are important to handle in a version n+1 of the propbank (supported by version n+1 of the grammar) are the case of the sentences that have their parse trees changed and the case of the new annotated sentences that entered the treebank. For the remaining propbanked sentences, whose annotation was not altered, they just need to be automatically transfered from version n to version n+1 of the propbank.

To support this dynamic propbanking, a tool was developed that performs the required comparisons between the annotated sentences in version n of the propbank and the outcome of version n+1 of the grammar applied over the pool of the sentence to be propbanked. Moreover, it singles out the sentences that need to receive the attention of the human annotators and transfers the remaining annotated sentences from version n to version n+1.

This comparator tool receives as input the spreadsheets S_n, of the previous version of the propbank, and the spreadsheets S_{n+1}, generated on the basis of version n+1 of the grammar. For each pair of spreadsheets containing the same suite of sentences, every sentence will be checked.

If a sentence maintains its parse tree from version n to version n+1, its propbank annotation will be maintained and transfered from S_n to S_{n+1}. Thus, the annotators do not have to re-annotate these sentences.

If in turn a sentence did not have a parse in version n and received a parse in version n+1, its sheet is signaled in the spreadsheet by means of a designated format for its name. Also, if a sentence receives a parse tree in version n+1 that is different from the parse tree in S_n, its sheet is also signaled. Additionally, for the sake of documentation, the parse tree and its semantic role labels in S_n are copied into S_{n+1} to a position in the corresponding sheet below the new parse tree whose propbanking is to be completed by the annotator.

4 Propbanking with a deep linguistic grammar

The practical viability of the propbanking environment just described was tested in a pilot experimental study. In this study, this environment was used to propbank 807 sentences (5,457 tokens overall; longest sentence with 34 words), extracted from a corpus of newspaper texts with a total of 350 Ktoken and 12 Ksentence [1]. This is an annotated corpus accurately annotated with part-of speech and morphological information and these annotations were used to help constrain the grammar search space.

The propbanking was performed under the annotation methodology consisting in a double blind annotation followed by adjudication. The annotation was produced by two annotators, who hold a degree in the Humanities with formal training in Linguistics and who have more than one year of experience in corpus development and treebanking. The adjudication was done by a third element of the team, who holds a degree in Linguistics and was the main coder of the grammar used.

The 807 sentences (5,457 tokens) were propbanked in 10 hours by the annotators, without any previous period of adaptation to the task in this study. This indicates that a rate of at least 80 sentences (550 tokens) propbanked per hour can be expected, provided these sentences had been previously treebanked in this overall environment for corpus development. A propbank with a typical size of 1 Mtoken could thus be expected to be produced out of a deep linguistic treebank in about 2,000 hours, with an estimated 100 person month of effort for a double blind annotation with adjudication.

This study also allowed getting a first indicative assessment of the level of reliability of the data produced by the annotators, before adjudication, that can be expected for a proposition bank produced under this methodology. The inter-annotator agreement coefficient used was Cohen's κ coefficient [8], calculated by the formula

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

where A_o is the observed agreement, and A_e is the expected agreement. Expected agreement is the sum of the expected agreement for every tag, where the expected agreement for each tag is the probability of the two annotators to assign that tag by chance. The probability of one annotator assigning a given tag by chance is the proportion of items actually assigned by that annotator with that tag.

The score for the inter-annotator agreement was 0.77 for an observed agreement of 0.83 and an expected agreement of 0.27, calculated over a total number of 334 assigned tags (tokens).

5 Conclusions

Annotated corpora tend to include increasingly sophisticated linguistic information. When developing treebanks with deep linguistic representations, the linguistic information to be associated with each sentence is so complex that it cannot be safely done manually and the annotation has to rely on some computational grammar supporting it. In this paper we discussed how to extend the utilization of deep linguistic grammars as supporting tools also for the annotation of propbanks.

In order to explore the contribution of the grammar for propbanking to the maximum extent possible without hurting its efficiency, the propbanking environment studied was based on two phases. In a first phase, the grammar is used to treebank the sentence. As the grammatical representations produced by the grammar already include predicate-argument structures, this information was explored to annotate the argument phrases with the semantic role labels ARG1,. . . ,ARG*n* for arguments. In a second phase, the parse trees were converted to an interface format that singled out the modifier phrases, whose propbanking still needs to be completed by the human annotators.

As the grammar is typically an application that will be extended and refined over time, the propbanking environment needs to be prepared to support the construction of a dynamic propbank. This is achieved with the help of a comparison tool that permits drawing the attention of the annotators only for the sentences newly entered in the treebank.

This propbanking environment was tested in a preliminary experimental study. In this study, though the annotators and the adjudicator were performing their tasks for the very first time, without previous substantial training that could be gathered after continued work, the results of this pilot study were very promising. They suggest that, provided human annotators have been sufficiently exercised over a large enough portion of corpus, this environment will support the propbanking of over 80 sentences per hour and permits to expect a reliable level of inter-annotator agreement, that will rise over 0.77 in terms of the κ coefficent. A corpus with a large size of 1 Mtoken could thus be expected to be produced out of a deep linguistic treebank with less than 100 person month of annotation effort with a double blind annotation methodology.

References

[1] Florbela Barreto, António Branco, Eduardo Ferreira, Amália Mendes, Maria Fernanda Nascimento, Filipe Nunes, and João Silva. Open resources and tools for the shallow processing of portuguese: the TagShare project. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, 2006.

[2] Emily M. Bender, Dan Flickinger, and Stephan Oepen. The Grammar Matrix: An open-source starter-kit for the development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Procedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan, 2002.

[3] António Branco and Francisco Costa. Accommodating language variation in deep processing. In Tracy Holloway King and Emily M. Bender, editors, *Proceedings of the Grammar Engineering Across Frameworks Workshop (GEAF07)*, pages 67–86, Stanford, 2007. CSLI Publications.

[4] António Branco and Francisco Costa. Self- or pre-tuning? deep linguistic processing of language variants. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 57–64, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[5] António Branco and Francisco Costa. A computational grammar for deep linguistic processing of portuguese: LXGram. In *Technical Reports Series*. University of Lisbon, Department of Informatics, 2008.

[6] António Branco and Francisco Costa. LXGram in the shared task ”comparing semantic representations” of STEP2008. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing*, pages 299–314, London, 2008. College Publications.

[7] Jean Carletta, Amy Isard, Stephen Isard, Jacqueline C. Kowtko, Gwyneth Doherty-Sneddon, and Anne H. Anderson. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13–32, 1997.

[8] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[9] Ann Copestake. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, 2002.

[10] Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. Minimal Recursion Semantics: An introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332, 2005.

[11] Lynette Hirschman, Patricia Robinson, John Burger, and Marc Vilain. Automating coreference: The role of annotated training data. In *Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 1997.

[12] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[13] Stephan Oepen and John Carroll. Parser engineering and performance profiling. *Natural Language Engineering*, 6(1):81–98, 2000. (Special Issue on Efficient Processing with HPSG).

[14] Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 1253–7, Taipei, Taiwan, 2002.

[15] Martha Palmer, Dan Gildea, and Paul Kingsbury. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1), 2005.

[16] Carl Pollard and Ivan Sag. *Information-Based Syntax and Semantics, Vol. 1*. CSLI Publications, Stanford, 1987.

[17] Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. Chicago University Press and CSLI Publications, Stanford, 1994.

Annotation of Selected Non-dependency Relations in a Dependency Treebank

Kristýna Čermáková, Lucie Mladová, Eva Fučíková, Kateřina Veselá

Charles University in Prague
Institute of Formal and Applied Linguistics

E-mail: {cermakova, mladova, fucikova, vesela}@ufal.mff.cuni.cz

Abstract

The following paper has two aims. First, it introduces a procedure of a manual annotation of selected linguistic phenomena across a large-scale dependency treebank of English. The method was designed to provide higher consistency of annotated data, and so higher credibility of the treebank. Second, the first expert task completed by means of this method is being described – the annotation of rhematizers and discourse connectives and their modifiers, i.e. annotation of some non-dependency relations in a dependency approach.

1 Motivation

Disagreements between annotators' judgments in corpora annotation are a disturbing factor in the machine training. Nevertheless, they represent an important indicator of functionality of the used approach; they localize theory deficiencies, and are also useful for finding interesting language phenomena.

Prior to the first release of the Prague English Dependency Treebank (PEDT) [2], a set of control scripts was established to increase consistency of the annotated data. Since the PEDT adopted its annotation scheme and guidelines from its Czech "mother treebank" – the Prague Dependency Treebank (PDT) [7], we expected some of the repeated disagreements to be caused by different treatment of language-specific phenomena in the two corpora, and therefore to be solved by adding new, language-specific annotation rules. Hence, the most frequent disagreements in PEDT were identified, analyzed, and rectified through additional manual annotation. The control scripts revealed that the divergences had been partly a matter of capturing conventions; partly they were difficult linguistic issues that required deeper linguistic knowledge. The former was handled by introducing automatic changes [8], and the latter was solved by the development of a new annotation method, the so called expert annotation. In this paper, we offer the description and evaluation of this specific manual annotation procedure.

2 Prague English Dependency Treebank

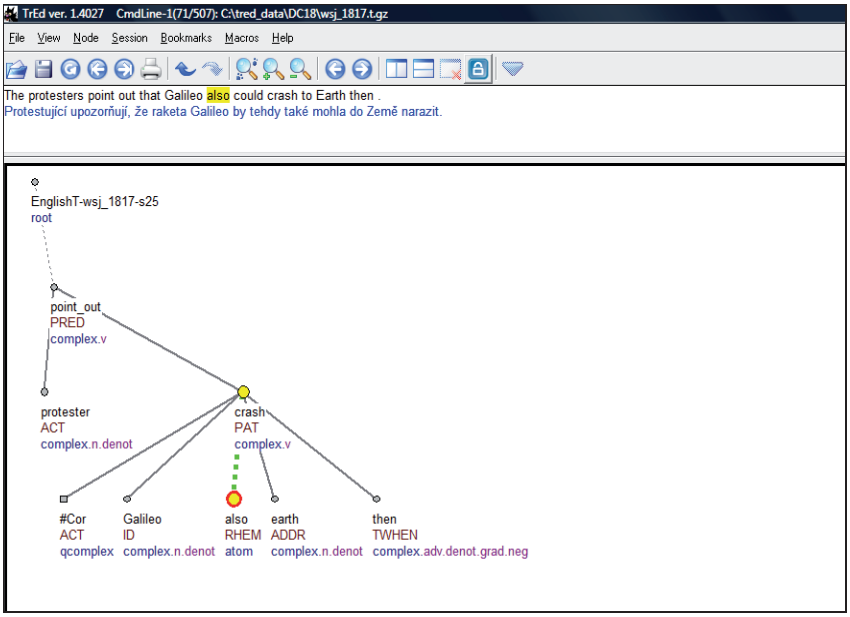
The Prague English Dependency Treebank (PEDT) represents the English-language part of the Prague Czech-English Dependency Treebank (PCEDT, [1]). PCEDT is a parallel corpus developed by Czech linguists primarily for the purpose of experiments in machine translation with a special emphasis on dependency-based (structural) translation. PCEDT and therefore also PEDT are based on the long-standing Praguian linguistic tradition and the Functional Generative Description of language (FGD) [6], adapted for the current computational linguistics research needs. The first publicly released version PEDT 1.0 [2] comprises annotation of approximately 25% of all approximately 49 000 sentences from Penn Treebank III – the Wall Street Journal section.

Wall Street Journal texts in PEDT are manually annotated on the tectogrammatical layer which represents underlying syntactic structure and captures semantic relations. The tectogrammatical layer annotation comprises dependency structure in the form of a dependency tree, including semantic labeling (the so called functors), valency annotation, and some coreference relations. Currently, routine annotation proceeds (approximately 50% of the data have already been annotated), and annotation principles are refined. The goal of the project is to annotate the entire PTB III – WSJ. Simultaneously with the annotation of English data, Czech translations are annotated giving rise to the parallel PCEDT.

3 Annotation of Specific Phenomena

The standard manual annotation of the tectogrammatical layer in PEDT proceeds since late 2006 on the dependency-converted tree structures. The division of the data into the original WSJ sections is preserved, with each annotator receiving one section at a time to be examined tree by tree. The inter-annotator agreement is measured regularly on a subset of simultaneously annotated data. For the two main attributes in the treebank, i.e. structure and functor, the agreement ranges from to 81 to 91% (functor) and 90 to 96% (structure) with a slight rising tendency.

A specific procedure was proposed to solve especially problematic syntactic issues, such as the treatment of some of the non-dependency edges described further in Section 4 of this paper. First, the nature of the problem was identified; second, the PEDT was scanned and sentences with the occurrence of the problematic phenomenon (problematic lexemes, phrases or functors) were selected and located into filelists. Particular questionable parts (nodes of the trees) within the filelists were highlighted in the annotation tool (see Figure 1) [4], and finally, “expert” annotators trained for the given linguistic task examined the filelists across all corpus sections. Where possible, correct analysis was pre-annotated automatically, for instance the correct lemmatization of typically unambiguous multiword expressions such as *as a result*, *for example* or *in other words*.



The protesters point out that Galileo also could crash to Earth then .
Protestující upozorňují, že raketa Galileo by tehdy také mohla do Země narazit.

Figure 1: A highlighted node for the expert annotation in the tree editor TrEd

4 Non-dependency Relations in PEDT

Based on the tradition of FGD, PEDT is a dependency corpus. However, it is able to convey various non-dependency relations by means of non-dependency edges which are established to represent primarily parataxis and some other specific relations. In our specific phenomena annotation, we focused on the problematic non-dependency edges that are used to capture proposition modifiers (expressions with a lesser degree of integration into the syntactic structure, such as modal and attitude markers, focusing and additive expressions, etc.). In PEDT, these are roughly reflected by nodes with the semantic role of:

- expressions referring to preceding contexts (PREC),
- rhematizers (RHEM),
- conjunction modifiers (CM),
- and marginally attitude (ATT).

Nodes with the functor (semantic label) PREC function as discourse connectives. Basic forms of these linking expressions are adverbials (*consequently*), particles (*yet*), some prepositional phrases (*in addition*), and paratactic connectives (*therefore*). From the point of view of traditional English grammars [5], they are partly homonymous with verbal complements, most often with temporal and spatial ones. If they assist in connecting paratactically conjoined elements, they are usually assigned the functor for conjunction modifiers (CM).

The nodes with the functor RHEM function as rhematizers, i.e. expressions whose function is to signal the topic-focus articulation categories in a sentence, namely the communicatively most important categories – the focus and contrastive topic [3], and their scope is indicated by a non-dependency edge. Rhematizers, e.g. *even, just, solely, exactly, precisely, only, alone, merely, simply, especially, particularly, in particular* resemble adverbials but they differ from them by their ability to modify not only a verb and adjective but also a syntactic noun. Additive expressions such as *too, also, again, equally, similarly, likewise, as well, in addition* rank sometimes among rhematizers although their function is primarily connective.

The first specific phenomena annotation completed throughout the PEDT is the annotation of the described semantic groups, mainly PREC and RHEM.

5 Rhematizers and Discourse Connectives in PEDT

In this section, we describe some of the repeatedly occurring problems concerning non-dependency edges from the linguistic point of view. Within the group examined, there is a huge functional homonymy among various uses of the same lexical unit. Sometimes, only a larger context and its analysis from the point of view of topic-focus articulation and/or discourse structure are needed to interpret the function of a particle or adverbial correctly. However, in certain cases, even with substantial background knowledge, an unambiguous solution is not to be found. For our annotation it was crucial to distinguish between the cases in which the given expression had its original adverbial meaning (i.e. there was a proper dependency relation), and the cases in which it functioned otherwise (i.e. as a node with a non-dependency edge).

5.1 Rhematizers and Extent Adjuncts

One of the most disputed problems is the homonymy of expressions with the semantic component of extent in their meaning. If they express the extent or degree, they are interpreted as extent adjuncts, e.g. (1a). But, if the interpretation allows also the "*primarily*" meaning, as possibly in (1b), it can be treated both as a syntactic member with the semantic role of extent or as a focalizing element.

(1a) *He has cancelled numerous campaign appointments and was <u>largely</u> inaccessible to the media until the stock story broke.*
(1b) *The enormous inflation over the past 30 years was <u>largely</u> due to monetary policy.*

Another problematic group is represented by "typical" rhematizers, e.g. (2a). However, it should be noted that if such "typical" rhematizers modify a numeral or another quantitative expression, e.g. (2b) they act precisely like regular extent modifiers without a rhematizing function, cf. (2c):

(2a) *We invited <u>only</u> friends.*
(2b) *We invited <u>only</u> five friends.*

(2c) *We invited exactly five friends.*

In (1b) and (2b), both interpretations are correct depending on the point of view of the analysis. Yet, for the purpose of semantic labeling in PEDT, a uniform rule for such cases had to be established. Therefore, (1b) was treated as a rhematizer and (2b) as an adjunct of extent.

5.2 Rhematizers and Discourse Connectives

Rhematizers relate to a smaller or larger part of a clause, i.e. they can have a narrower or wider scope. Rhematizers with a narrow scope are easy to recognize, the only problematic usage of rhematizers in English is their position right before the verb, e.g. (3a), and (3b). With regard to the preceding context, annotators are able to distinguish between a narrow scope, e.g. (3a) and a wide scope of the rhematizer, e.g. (3b). Hence, only a larger context can help determine the scope of the rhematizer, and recognize whether the rhematizer relates to elements that precede and/or follow in the surface word order.

Further, if the rhematizer has an additive meaning and stands before the verb as in (3b), it can coincide with the function of a discourse connective. Discourse connectives, unlike rhematizers, relate always to two arguments, they connect two text spans. When a sentence-initial *also* is separated by a comma, it is treated always as a discourse connective (3c). The difference between (3b) and (3c) is considered formal, not semantic. Therefore, *also* in (3b) can be treated both as a rhematizer (RHEM) and a discourse connective (PREC) with equal validity. The type of sentences as in (3b) caused major problems in the expert annotation, also because of its high frequency in the treebank texts.

(3a) *Crude oil for November delivery edged up by 16 cents a barrel to $ 20.75 a barrel. Heating oil prices also rose.*

(3b) *The complex restructuring transforms London-based WCRS from primarily a creator of advertising into one of Europe's largest buyers of advertising time and space. It also creates a newly merged world-wide ad agency controlled by Eurocom.*

(3c) *The company said a drop in activity in the powerboat industry reduced sales volume at its two marine-related operations. Also, the company said its commercial products operation failed to meet forecasts.*

6 Inter-annotator Agreement

Three annotators who also have long-term experience with standard annotation of PEDT were trained for the specific task, and they annotated approx. 3000 problematic structures each. 515 structures were annotated by all three of them as a set of data for IAA measurement. The measurement itself is derived from the basic IAA measurement script for standard annotations [8], and it proceeds roughly as follows: Within the set of nodes either highlighted or touched by any of the three trained annotators, agreement regarding four attributes was computed between each pair of annotators. The attributes are the following: **structure** (parent node), **functor**, **tectogrammatical lemma** of the node, and **a/aux.rf**, i.e. links to the lower layer of surface syntax. The results are summed up in the Table 1.

Attribute/ Annotator pair	Structure	Functor	T-lemma	A/aux.rf
A x B	91.2%	91.1%	98.9%	96.2%
B x C	90.6%	89.6%	98.9%	92.7%
C x A	92.1%	89.3%	99.1%	93.6%

Table 1: IAA Measurement for specific phenomena annotation in the PEDT

In terms of expert annotation as such, the results can be considered satisfactory. They show that the method applied supported a unified approach to the phenomenon of focusing and additive expressions. The results are slightly higher for the same two annotator pairs compared to their recent results in standard, much more complex annotation. They also prove that the most difficult task for annotators (both in standard and specific phenomena annotation) is the agreement in functor. To sum up, before the implementation of the expert annotation, the most frequent instances of inter-annotator disagreement were caused by the lack of precise guidelines which allowed more interpretations. In the expert annotation, such disagreements occur extremely rarely.

7 Conclusion and Future Work

Our specific annotation proved to be an effective solution of the annotation of complicated phenomena. It eliminates mistakes, and simultaneously does not inhibit the standard annotation from proceeding. We were able to refine more than 9000 sentences, which is approx. 19% of the treebank. As we expected, the problematic issues examined such as the distinction between the rhematizing, discourse linking and simply adverbial function of homonyms or defining the scope of a rhematizer are too complex to be currently successfully resolved merely by automatic annotation.

The new annotation method demonstrated a significant difference between English and Czech not only in terms of standard word order principles (as is generally known) but also in terms of rhematizer positioning principles. On the one hand, English word order is largely determined by grammatical principles and as such it displays less flexibility than Czech word order. On the other hand, rhematizer positioning in English is far more flexible than it is in Czech. The data concerned in the specialized annotation show that English (unlike Czech) is able to place rhematizing expressions on the border between the topic and focus notwithstanding their distance (in the linear surface word order) from the focus proper, i.e. the informationally most weighted element. Anyway, it was not the surface position but the scope of pre-verbal rhematizers that was most problematic issue even for trained annotators.

Based on the positive results of the fist specific phenomena annotation, another run is in preparation which will focus on the annotation of complicated comparative structures.

8 Acknowledgements

The research this paper is based upon was supported by the following grants: GAUK 103609, GA201/09/H057, GA405/09/0729, MŠMT ČR LC536, and EU – Companions.

References

[1] Čmejrek, Martin, Jan Cuřín, Jiří Havelka, Jan Hajič, and Vladislav Kuboň. 2005. Prague Czech-English Dependency Treebank. In *EAMT 2005 Conference Proceedings*, p. 73–78.

[2] Hajič, Jan et al. 2009. *Prague English Dependency Treebank 1.0*, Software or data, Institute of Formal and Applied Linguistics, Charles University in Prague.

[3] Hajičová, Eva, Barbara Partee Hall, and Petr Sgall. 1998. *Topic-focus articulation, tripartite structures, and semantic content*. Boston: Kluwer Academic Publishers.

[4] Pajas Petr, Štěpánek Jan. 2008. Recent Advances in a Feature-Rich Framework for Treebank Annotation. In *The 22nd International Conference on Computational Linguistics – Proceedings of the Conference*, Manchester, p. 673–680.

[5] Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, Jan Svartvik. 2004. *A Comprehensive Grammar of the English Language.* London: Longman.

[6] Sgall, Petr, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Prague: Academia.

[7] Šindlerová, Jana, Lucie Mladová, Josef Toman, Silvie Cinková. 2007. *An Application of the PDT-Scheme to a Parallel Treebank.* In: NEALT Proceedings Series, Vol. 1, Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories, Bergen, Norway, p. 163-174.

[8] Toman, Josef. 2009. *Automatická anotace angličtiny na tektogramatické rovině (Automatic Annotation of English on the Tectogrammatical Layer*). Master's thesis. Charles University in Prague.

Dependency Annotation for Learner Corpora

Markus Dickinson

Indiana University
Bloomington, IN 47405
E-mail: md7@indiana.edu

Marwa Ragheb

Indiana University
Bloomington, IN 47405
E-mail: mragheb@indiana.edu

Abstract

Building from the CHILDES dependency annotation scheme and on interlanguage POS annotation, we describe a syntactic annotation scheme developed for the data of second language learners. We encode subcategorization frames and underlying dependencies, in addition to the usual surface dependencies. The annotation scheme is relatively independent of language and can be mapped to learner errors.

1 Introduction and Motivation

A prominent area of research in the collection of corpora containing the language of second language learners has been in annotating so-called errors (see, e.g., Granger, 2003). This helps further studies in computer-assisted error analysis and develop technology for detecting learner errors, but does not address more general properties of learner language, such as fluency and complexity (e.g., Pendar and Chapelle, 2008), stage of acquisition (e.g., Pienemman, 1998), or even basic features such as tense and aspect (e.g., Wulff et al., 2009). There is thus a need for annotated data which more generally supports second language acquisition (SLA) research.

What needs to be described is *interlanguage*, the in-progress language of learners which is a linguistic system in its own right, without focusing on errors. Díaz-Negrillo et al. (2009) annotate interlanguage part-of-speech (POS) properties, but, as far as we know, no one has syntactically annotated interlanguage. This is despite the fact that there is much work on automatically detecting learner syntactic errors (see, e.g., Vandeventer Faltin, 2003).

At the same time, any such annotation effort would ideally also be useful to the computational linguistics community. To accurately parse learner data, it is desirable to have a collection of sentences with proper annotation for evaluation. To that end, we describe work in developing an annotation scheme to encode syntactic relations in learner language.

2 Background

2.1 POS annotation of learner data

Díaz-Negrillo et al. (2009) address the notion of part-of-speech (POS) annotation for interlanguage, distinguishing three types of evidence needed to capture it: 1) stem/lexicon lookup, 2) morphological cues, and 3) word distribution. Because a learner does not always use a word in a native way, these pieces of evidence can conflict, and thus a POS annotation scheme which conflates all three, such as one developed for native language, cannot properly annotate learner language. For instance, in (1) (=(15) in Díaz-Negrillo et al. (2009)), there is a mismatch in the word's morphology (past tense) and its distribution (past participle).

(1) it has **grew** up a lot specially after 1996

While this serves as an excellent starting point, only POS annotation is explored, and this is done by examining an error-annotated corpus and seeing what POS information is necessary to account for the different error types. We start in the opposite direction, assuming no error annotation, and we develop a syntactic dependency annotation scheme for learner language.

2.2 Parsing of learner data

Although learner corpora have not generally been annotated with syntax, there is much work on syntactically parsing learner data (e.g., Vandeventer Faltin, 2003), to detect and diagnose ill-formed structures, such as non-agreement between a subject and a verb. For example, Menzel and Schröder (1999) use weighted constraints to derive dependency structures for learner language, and constraints which are violated are used to recover error information about the sentence. The sentence is robustly parsed, so the resulting dependency structure in some sense captures interlanguage.

However, it is not clear what exactly the surface syntax is encoding, as the parse is based on a model of native language. Further, it is unlikely that surface dependencies (or constituencies) capture the full set of syntactic facts employed by a learner. We return to these issues in section 4.1 after outlining general properties of our annotation scheme, including POS.

3 Annotation Scheme

For our pilot annotation, we use a collection of essays, from learners of different levels, from the early 1990s. The learners watched a short cartoon (*Tin Toy*) and were asked to discuss what happened. Examining this data has revealed several major theoretical issues for annotating second language data, which we outline in this paper.

3.1 General principles

In developing an annotation scheme, we followed several general guiding principles. The first principle is to annotate the language *as is*, i.e., annotate only what is there. We want to make as few claims as possible about what the intended meaning of the learner is, aiming only at an adequate description of the learners' interlanguage, from which researchers can draw their own conclusions. Thus, we do not posit empty elements or corrected forms, and errors do not exist directly in the corpus (though, see section 5). Further, since we build from previous annotation schemes, if a particular tag requires us to infer properties about the learner which we cannot possibly know, we prefer to adjust the annotation scheme.

Secondly, we try to give the learner the benefit of the doubt. For example, in (2), it is not entirely clear how the learner is trying to use *fraid*. We mark it as being a predicate, with *toy* as its subject, and do not attempt to posit any particular non-native properties, such as marking *(a)fraid* as a verb.[1]

(2) The toy **fraid**.

Since we stay as close to the original text as possible, we do not resegment run-on sentences. However, since these sentences have multiple syntactic roots (see section 4.2), they can easily be divided into smaller units.

3.2 Basic annotation

To make the corpus maximally useful for searching, we include lemma information for each word. This normalizes a word across all its realizations, including spelling mistakes. As we will see, it also allows us to capture properties akin to the stem information in Díaz-Negrillo et al. (2009).

We then annotate the corpus using the SUSANNE tagset (Sampson, 1995), as it distinguishes properties potentially of interest for SLA research, e.g., transitive and intransitive verbs, countable and uncountable nouns, and definite and indefinite articles. We have split the POS annotation into two parts, however. Namely, we have one POS tag to refer to the linguistic *form* of a word, which generally refers to its morphological features, and another POS tag to refer to the syntactic *use* of a word. This is essentially the same as the distinction between morphological and

[1]We still capture the fact that the text is non-native by encoding an empty subcategorization list, as with any other predicative (with or without a copula); see section 4.3.

distributional evidence in Díaz-Negrillo et al. (2009). In example (3), for instance, *makes* has the morphological form of 3rd person singular present tense (VVZt), but, in its position following *can*, its use is as a baseform verb (VV0t).

(3) Tin Toy can **makes** different music **sound**.

We leave a tag underspecified if it is not clear how the token is being used, as we can see for the word *sound* in (3). The form is clearly singular (NN1c), but the learner may be using this form as either a singular or plural noun (NN1c or NN2). Therefore, the *use* tag is underspecified (NN).

Where we differ from Díaz-Negrillo et al. (2009) is in not encoding a *stem* tag. This is potentially problematic, as the discrepancies between the inherent stem properties and the morphology account for realizations such as *choiced* in (4) (=(9) in Díaz-Negrillo et al. (2009)). However, we have recourse to the lemma information, which is essentially a stem lookup. One possibile solution in this particular case is to make the lemma *choose* (instead of *choice*), with the form and use tags as participles.

(4) . . . to be **choiced** for a job . . .

New & redefined tags We have had to make some changes to the tagset to account for learner language. Firstly, we use compound tags for words which have been merged. In (5), for instance, we have the POS use tag of AT1+NN1c for *adram*, which seems to be a blend of *a drum*.

(5) The tin toy had **adram** and a acordion.

Secondly, in the interest of underspecification, we have added some tags. With verbs, for instance, there are contexts where more than one tense is possible, as in (6). In this case, we do not know the specific POS use of *follow*; we only know that it is tensed.[2] The SUSANNE tagset specifies particular tense properties (e.g., D for past tense); to bypass making this decision, we define an underspecified VVTt, where T stands for a tensed verb.

(6) The child **follow** him.

3.3 Annotation format

We annotate the corpus with the standard CoNLL format, as in figure 1, which allows for dependency relations (Buchholz and Marsi, 2006). Thus far, we have described columns 1-5 (id, word, lemma, form POS, use POS); the remaining columns are described in the next section.

[2]The context of the video can help disambiguate some learner ambiguities, but the annotation scheme is necessary in cases where it cannot.

```
1 Tin       tin       NP1x NP1x _                  2 MOD   _ _
2 Toy       toy       NP1x NP1x _                  4 SUBJ  _ _
3 can       can       VMo  VMo  _                  4 AUX   _ _
4 makes     make      VVZt VV0t <SUBJ, AUX, OBJ> 0 ROOT  _ _
5 different different JJ   JJ   _                  7 MOD   _ _
6 music     music     NN1u JJ   _                  7 MOD   _ _
7 sound     sound     NN1c NN   _                  4 OBJ   _ _
8 .         .         YF   YF   _                  4 PUNCT _ _
```

Figure 1: Example CoNLL format

4 Dependency Relations

4.1 Capturing interlanguage syntax

We mark dependency relations between words, as this annotation captures many grammatical properties relevant to language acquisition, such as agreement of morphosyntactic features (e.g., Parodi, 2000) and argument structure (e.g., Mellow, 2008). It is also multi-lingual, making the scheme adaptable for use with other languages and readily available for dependency parsing (see, e.g., Buchholz and Marsi, 2006). Furthermore, encoding dependencies can be done more quickly than with constituencies.

Orthogonal to the decision to encode dependency relations, however, we must ask what types of evidence need to be brought to bear on learner language. What evidence determines the syntactic usage of a word, especially when that usage may be non-native? Consider, for instance, the constructed example (7). We know *He* in (7a) is the subject because of the morphological case marking of nominative, and the syntactic distribution indicates that nominals preceding verbs are possible subjects.

(7) a. **He** wants to save his life.
    b. **Him** wants to save his life.

We thus want to consider **distributional** evidence in encoding syntactic properties. Languages vary in to what degree distribution is determined by word order and to what degree by morphological markings. For English, we place a greater emphasis on word order, or positional, information for determining grammatical relations, for a few reasons. First, English case marking is very deficient, found only on pronouns. Secondly, with a solid POS platform, morphological discrepancies are largely already accounted for: in (7b), for instance, we have a accusative POS form with a nominative POS use. Thirdly, as we will see below, when the morphology conflicts with the word order, this can be accounted for via other means. The syntactic distributional evidence we propose to use is essentially what is encoded by a surface dependency annotation scheme.

As a second piece of evidence, we need to consider inherent lexical properties of a word. Consider the difference between the constructed examples (8a) and (8b): in both cases, *couch* is acting as an object of the verb. The difference is in the **subcategorization** properties of the two verbs: *owns* selects for a subject and object (<SUBJ,OBJ>), while *snores* only selects for a subject (<SUBJ>). While words can be ambiguous in their subcategorization frames, there is generally only one acceptable for a given sentence.

(8) a. He **owns** a couch.
    b. He **snores** a couch.

Consider now the real learner example in (9). The word *dull* (assuming its intended form *doll*) is ambiguous: it could be an object of *escape* (with a missing subject), or it could be the subject in the wrong location. By biasing the distributional evidence towards positional information, we can say that *dull* is an object, but it seems like the learner simply misplaced the subject.

(9) . . . escape the dull [doll]

One could underspecify the dependency labels (e.g., Carroll et al., 2003)—e.g., ARG instead of SUBJ/OBJ—but the issue is really that of putting together a complete argument structure for this verb. Giving the learner the benefit of the doubt, they have all the elements to form a well-formed semantic meaning from this sentence (cf. *the doll escaped*). There is a notion of what we will call **underlying dependencies** to account for. Using the argument structures and semantics of the words in the sentence as evidence, all the argument slots can reasonably be filled. This is in line with approaches mapping to underlying functor-argument structures (e.g., Sgall et al., 2004; Kromann, 2003).

In example (9), for instance, the underlying dependency of *dull* is as a subject of *escape*, even though the surface dependency is an object. A *dull* (*doll*) is an acceptable syntactic and semantic argument of *escape*, and so this is the relation we want if the goal is to be able to create a syntactically and semantically well-formed structure.

Underlying dependencies are useful not only for encoding learner ambiguities, but also long-distance dependencies. Given the generally-held assumption that each word has only one head in a dependency graph, surface dependencies cannot capture every relationship. Consider the constructed example (10a), for instance. Here, the verb *owns* is in a relative clause, with its subject immediately on the left. The verb is transitive, however, with its object left-dislocated, and surface dependencies do not account for this. This is a crucial piece of information, when one considers an intransitive verb such as *snores*, as in (10b), where both verbs have a long-distance object (*couch*).

(10) a. The couch that he **owns** burned.
     b. The couch that he **snores** burned.

One final piece of evidence is that of **collocations**, where one word selects another. In (11), for example, *interest* collocationally restricts the range of following prepositions, making the string *interest with* sound non-native.

(11) The baby had no more interest **with** the tin toy.

4.2 Encoding distributional (surface) dependencies

With the CoNLL format, it is straightforward to encode surface dependencies: columns 7 and 8 mark the head of each word and the relationship. The surface dependency tree for figure 1, for example, is shown in figure 2.[3]

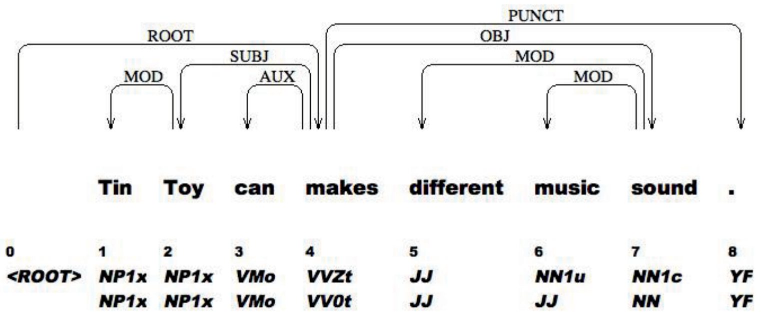

PUNCT
ROOT
OBJ
SUBJ
MOD
MOD
AUX
MOD
Tin Toy can makes different music sound .
0 1 2 3 4 5 6 7 8
<ROOT> NP1x NP1x VMo VVZt JJ NN1u NN1c YF
NP1x NP1x VMo VV0t JJ JJ NN YF

Figure 2: Example dependency tree

The scheme used to mark these relations is the one used for the CHILDES database of first language learners (Sagae et al., 2004, 2007). This was chosen because it was developed for language "in progress," and it encodes fairly specific grammatical relations, a total of 37 relations. Furthermore, if needed, the scheme can be mapped to a scheme which allows for underspecification of grammatical relations (Carroll et al., 2003).

New & redefined labels Despite the usefulness of the CHILDES annotation scheme, we have had to make some minor alterations. For example, we have added a label APPOS, to account for appositions, such as in (12), where *and* is an APPOS dependent of *things*.[4]

(12) thir are two *things* besaid the Toy, raings **and** string of beads.

[3]We use MaltEval (Nilsson and Nivre, 2008) for displaying trees.

[4]In coordinate structures, *and* is the head, a decision we follow for convenience.

Secondly, we have clarified the use of X and C before various labels (SUBJ, COMP, PRED, JCT). According to the documentation,[5] C indicates a finite clause and X a non-finite clause, which usually corresponds to C clauses having a subject and X not. In our data, however, a subject sometimes accompanies a non-finite verb or there might be no verb, so neither label technically fits the data we have. In (13), for example, *save* is some type of COMP of *want*, but it is not clear what its finiteness properties are.

(13) ... he *want* **save** his life.

We therefore re-define X and C categories, to focus on the presence or absence of subjects: C indicates the presence of a subject and X the absence. This is more straightforward to determine in English learner data than verb tense, and our goal is to make as few decisions as possible. This definition also seems to be in line with the intention of the annotation scheme (see the discussion of XCOMP parser errors in Sagae et al., 2007, sec. 5).[6]

Underspecifying information A full decision cannot always be made for the dependency graph. In (14), for example, we find the extraneous word *at*, and it is not clear where to put it in the dependency tree. In such cases, we attach the word to the virtual root and assign it no dependency label.

(14) He begins to walk and **at** to run.

4.3 Encoding subcategorization

We encode subcategorization frames in slot 6 of the CoNLL files, a spot for "other syntactic and morphological features." This is encoded as a list of dependency relations, e.g., <SUBJ, OBJ>. The only encoded elements are selected arguments and not adjunct dependents. If a word selects for nothing, we use '_' to indicate an empty list (<>).

Since cases are ambiguous, we encode the closest possible match. For *the toys*, for example, the subcategorization value of *toys* is <DET>, while in the bare NP *toys*, it is _. When a case is not clear, we give the learner the benefit of the doubt. For instance, in (15) (=(3)), the subcategorization for *sound* could be either <DET> or _, depending on if the learner intended to use a singular or plural noun. We label it _, with the number discrepancy already accounted for with POS (form=NN1c, use=NN).

(15) Tin Toy can makes different music **sound**.

[5]`http://www.cs.cmu.edu/~sagae/childesparser/childes-annotation.pdf`

[6]This is a fairly English-specific decision and would need to be adapted for languages which drop subjects; similar, consistent solutions can likely be found.

4.4 Encoding underlying dependencies

To encode underlying dependencies, we have decided only to encode those relations which are not already encoded at the surface level. Any relation already encoded as a surface relation is also interpreted as an underlying relation; if a relation covers the same pair of words as the surface syntax does, then the underlying relation overwrites the surface one (see (9)).

We encode this information in column 10 of the CoNLL files, using a list of pairs of head positions and relations. We cannot encode a single-headed dependency tree structure, as with surface relations, since a word may have several heads. For example, in (16), *I* serves as the subject of three different verbs, illustrated in figure 3: the surface subject of *hope* (7) and the underlying subject of *do* (9) and *enjoy* (12). The effect of this is also that the subcategorization lists for those verbs can be seen to be saturated.

(16) Now the only thing **I** *hope* to *do* is to *enjoy* him very well ...

```
1  Now   now   RTo    RTo    _                   10 JCT   _ _
2  the   the   AT     AT     _                   4  DET   _ _
3  only  only  JBy    JBy    _                   4  MOD   _ _
4  thing thing NN1c   NN1c   <DET>               10 SUBJ  _ <9,OBJ>
5  that  that  CST    CST    _                   7  CPZR  _ _
6  I     I     PPIS1  PPIS1  _                   7  SUBJ  _ <(9,SUBJ),(12,SUBJ)>
7  hope  hope  VV0t   VV0t   <SUBJ, XCOMP>       4  CMOD  _ _
8  to    to    TO     TO     _                   9  INF   _ _
9  do    do    VD0    VD0    <SUBJ, INF, OBJ>    7  XCOMP _ _
10 is    be    VBZ    VBZ    <SUBJ, XPRED>       0  ROOT  _ _
11 to    to    TO     TO     _                   12 INF   _ _
12 enjoy enjoy VV0t   VV0t   <SUBJ, INF, OBJ>    10 XPRED _ _
13 him   he    PPHO1m PPHO1m _                   12 OBJ   _ _
14 very  very  RG     RG     _                   15 JCT   _ _
15 well  well  RR     RR     _                   12 JCT*  _ _
```

Figure 3: A CoNLL example with underlying dependencies

Encoding all underlying dependencies in a single list preserves the CoNLL format. However, this can easily be mapped to a more principled representation, such as DeccaXML (Boyd et al., 2007), to allow multiple heads.

4.5 Encoding collocations

Collocational knowledge is encoded by affixing an asterisk to the relevant dependency relation when words are used non-natively. This allows us simply to mark awkward collocations without having to develop a robust theory of them. In figure 3 above, for example, we can see a slightly unnatural-sounding collocation of *enjoy well*, marked as a JCT* relation.

5 Mapping to errors

Our goal has to been to annotate learner language as it appears, but, to maximize the utility of the corpus, we want to be able to map our representation to one which is also useful for developing error detection systems or the study of learner errors (see, e.g., Ellis, 1994). We thus sketch here how mismatches between annotation levels point to errors.[7]

Form tag $\neq$ Use tag As mentioned previously, discrepancies between form and use POS tags can point to non-native usage. In (17), for example, *crawl* is form-annotated as VV0i (baseform verb) and, in this context, use-annotated as VVDi (past tense).

(17) A baby **crawl** around the room ...

Surface $\neq$ Underlying dependencies With both surface and underlying dependencies, we can identify those positions which conflict. For example, in (18), *baby* is marked as the surface object (OBJ) of *apparer*; the same positions have an underlying dependency of subject (SUBJ), however, pointing to some type of misuse. The issue here might be with word order or with argument structure; by annotating the layers separately, we point to the error without having to draw a particular conclusion about its nature.

(18) Then to *apparer* a **baby**.

Subcategorization $\neq$ Underlying dependencies Since the subcategorization lists indicate which arguments are selected for by a word, we can compare the list of potential arguments against those that are actually realized. We must check against the underlying dependents, not the surface dependents, because only in the underlying dependents do we find long-distance relations. This will capture sentences with missing, extra, or wrong arguments, as all are mismatches. An example of a missing argument is given in (19) (=(13)), where the infinitival *to* is missing. In this case, the subcategorization value of *save* is <SUBJ, INF, OBJ>, yet only the SUBJ and OBJ are realized (*he* and *life*, respectively).

(19) ... *he* want **save** his *life*.

To fully detect these errors, we have to know which types of dependents are arguments and which are adjuncts. For example, a JCT (adjunct) relation is a legitimate dependent of verbs, but is not selected. This has been clear-cut so far and can be ensured by splitting any ambiguous categories into separate argument and adjunct subcategories.

[7] We use the term *error* for any non-native usage, but ascribe it no theoretical status.

Limitations Currently, there are some aspects of learner language that we do not deal with, or in only limited ways. Anomalous word orders of adjuncts, for example, are not treated by our scheme.[8] In (20), for instance, the placement of *now* is odd. Currently, we mark such cases very cursorily, simply by adding a + sign on the dependency label (e.g., JCT+).

(20) He can't see **now** nothing.

Secondly, we do not handle semantic and pragmatic anomalies. In (21) (=(16)), for instance, *enjoy* should have probably been something like *entertain*. If it results in an anomalous argument structure, we will catch some of these errors indirectly, but otherwise we do nothing to mark them.

(21) Now the only thing I hope to do is to **enjoy** him very well ...

6 Summary & Outlook

We have introduced a new annotation scheme capturing dependency relations for learner language. Specifically, we saw the need for encoding subcategorization frames and what we have termed underlying dependencies, in addition to the usual surface dependencies. We also needed to make clear exactly what distributional properties surface dependencies relate to. This scheme allows us to encode learner language as it appears, but also allows the annotation to be mapped to errors.

The next steps are straightforward, starting with continuing to annotate our pilot data, refining the annotation scheme as needed, e.g., splitting up argument and adjunct labels and fleshing out word order issues. This process will be well-documented, leading to a set of guidelines, and we will also test inter-annotator agreement. We are now collecting new data which can be released publicly (so researchers can study, e.g., the most frequent mistakes).

Acknowledgements

We wish to thank Kathleen Bardovi-Harlig, Rex Sprouse, and Detmar Meurers for useful discussion, as well as three anonymous reviewers and the IU Computational Linguistics discussion group.

References

Boyd, A., Dickinson, M., and Meurers, D. (2007). On representing dependency relations – insights from converting the german tigerdb. In *Proceedings of TLT 2007*, pages 31–42, Bergen, Norway.

Buchholz, S. and Marsi, E. (2006). Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164, New York City.

[8]For argument word order, see the discussion on surface and underlying dependencies.

Carroll, J., Minnen, G., and Briscoe, T. (2003). Parser evaluation: Using a grammatical relation annotation scheme. In Abeillé, A., editor, *Treebanks: Building and using syntactically annoted corpora*, chapter 17, pages 299–316. Kluwer Academic Publishers, Dordrecht.

Díaz-Negrillo, A., Meurers, D., Valera, S., and Wunsch, H. (submitted, 2009). Towards interlanguage POS annotation for effective learner corpora in SLA and FLT.

Ellis, R. (1994). *The Study of Second Language Acquisition*. Oxford University Press, Oxford.

Granger, S. (2003). Error-tagged learner corpora and CALL: A promising synergy. *CALICO Journal*, 20(3):465–480.

Kromann, M. T. (2003). The danish dependency treebank and the underlying linguistic theory. In *Proceedings of TLT-03*.

Mellow, J. D. (2008). The emergence of complex syntax: A longitudinal case study of the esl development of dependency resolution. *Lingua*, 118(4):499 – 521.

Menzel, W. and Schröder, I. (1999). Error diagnosis for language learning systems. *ReCALL*, pages 20–30.

Nilsson, J. and Nivre, J. (2008). MaltEval: An evaluation and visualization tool for dependency parsing. In *Proceedings of LREC-08*, Marrakech.

Parodi, T. (2000). Finiteness and verb placement in second language acquisition. *Second Language Research*, 16(4):355 – 381.

Pendar, N. and Chapelle, C. (2008). Investigating the promise of learner corpora: Methodological issues. *CALICO Journal*, 25(2):189–206.

Pienemman, M. (1998). *Language Processing and Second Language Development: Processability theory*. John Benjamins.

Sagae, K., Davis, E., Lavie, A., MacWhinney, B., and Wintner, S. (2007). High-accuracy annotation and parsing of childes transcripts. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 25–32, Prague.

Sagae, K., MacWhinney, B., and Lavie, A. (2004). Adding syntactic annotations to transcripts of parent-child dialogs. In *Proceedings of LREC-04*, Lisbon.

Sampson, G. (1995). *English for the Computer: The SUSANNE Corpus and Analytic Scheme*. Clarendon Press, Oxford.

Sgall, P., Panevová, J., and Hajičová, E. (2004). Deep syntactic annotation: Tectogrammatical representation and beyond. In *Proceedings of the HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 32–38, Boston.

Vandeventer Faltin, A. (2003). *Syntactic error diagnosis in the context of computer assisted language learning*. Thèse de doctorat, Université de Genève, Genève.

Wulff, S., Ellis, N. C., Roemer, U., Bardovi-Harlig, K., and LeBlanc, C. (2009). The acquisition of tense-aspect: Converging evidence from corpora and telicity ratings. *The Modern Language Journal*, 93(3):354–369.

Linguistically Motivated Parallel Parsebanks

Helge Dyvik[♠], Paul Meurer[♡],
Victoria Rosén[♠♡], and Koenraad De Smedt[♠♡]

[♠]University of Bergen, Sydnesplassen 7, N-5007 Bergen (Norway)
[♡]Uni Digital, Allégaten 27, N-5007 Bergen (Norway)

{dyvik|paul.meurer|victoria|desmedt}@uib.no

Abstract

Parallel grammars and parallel treebanks can be a useful method for studying linguistic diversity and commonality. We use this approach to study how arguments to similar predicates are realized across languages. To that end, we formulate formal principles for aligning at phrase and word levels based on translational correspondences at predicate-argument level. A first version of a new tool for creating, storing, visualizing and searching treebank alignment at different levels has been constructed.

1 Introduction

A central concern within theoretical linguistics is the discovery of unifying patterns behind language diversity. Our aim is to study linguistic diversity and commonality through the use of parallel grammars and parallel treebanks. By parallel grammars we mean grammars for different languages constructed according to common principles, as in the ParGram project [3]. When parallel grammar writing is anchored in a common Lexical-Functional Grammar (LFG) framework with theory-imposed constraints [1], differences in grammatical analyses across languages are likely to reflect real differences between languages, rather than accidentally different descriptive strategies among grammarians.

A parallel treebank is a syntactically and possibly semantically analyzed parallel corpus, aligned not only on the word and sentence levels, but also on intermediate levels. Parallel treebanks, in which translationally corresponding phrases are linked, are a valuable (and still rare) resource, for example for research on innovative combinations of memory-based and knowledge-based machine translation.

Parallel treebank construction is a recently established and rapidly developing field, and it already includes experiments in automatic phrase alignment, notably Samuelsson and Volk [11]. Some problems arise from the fact that the syntactic structures in the treebanks to be aligned sometimes reflect different principles of analysis. Whereas Samuelsson and Volk's method starts from *n*-gram alignment (i.e. from the identification of translational correspondences between strings of words) to support phrase alignment, we want to explore the opposite direction: starting from correspondences between predicate-argument structures in a pair of sentence-aligned and tentatively word-aligned monolingual treebanks constructed from the two sides of a parallel (translational) corpus, we derive alignments at the phrase and word levels. In particular, we want to pursue the following hypothesis: *On the basis of monolingual treebanks constructed from a parallel corpus by means of parallel grammars, it will be possible to achieve automatic word and phrase alignment with significantly higher precision and recall than hitherto achieved through other means.*

In an LFG analysis a given f-structure (functional structure) is typically associated with more than one node in the c-structure (constituent structure). A set of nodes projecting the same f-structure is said to constitute a *functional domain*. Assuming an alignment of subsidiary f-structures, we expect that automatic phrase alignment can be achieved by alignments among the nodes in the functional domains of corresponding f-structures, according to criteria spelled out in 3.2 below.

Further assuming that the c-structures are organized according to common principles, the aligned phrasal categories are expected to be typologically informative. Our aim is to test these assumptions on typologically diverse languages: Norwegian, Dutch, Tigrinya and Georgian. Maximal c-structure diversity is guaranteed by the fact that these languages are spread out on the configurationality continuum — the configurational languages Norwegian and Dutch are at one end, and the free word order language Georgian is at the other [7], with Tigrinya in between [8].

In the current phase of our research, we are testing this approach on test suites especially constructed to bring out differences among the languages in the mapping of arguments to syntactic functions. Below we discuss the principles of our methodology, we propose formal alignment principles, and we present the first version of a tool that is unique in that we create, store, visualize and search correspondences between multiple languages at two syntactic levels, c- and f-structure, through a Web interface.

2 Methodology

An LFG-based parsebanking approach [9] is extended to multiple languages. The LFG framework is used because it is a substantial theory about the class of possible human languages, and not just a tool for grammatical description. While the basic projection formalism for codescription in LFG allows a wide range of c-structures to be associated with the same f-structure, and vice versa, including pairings that are implausible linguistically, recent LFG research has proposed strong universal constraints on the possible relationship between the two kinds of structures, implying empirical claims about the limits of possible variation among languages.

A central contribution to this research is Bresnan's development of a theory of constraints on the relationship between c-structures and f-structures [1]. By basing our grammars on Bresnan's proposals we intend to extend the notion of parallel grammars from just considering f-structure, as in ParGram, to encompassing c-structure as well. A consequence of adopting such common principles is that we will approach a situation where phrase-structural differences between analyses of translationally corresponding sentences will reflect genuine differences among the languages, and not just arbitrarily different principles of analysis among grammarians.

A fundamental task of a grammatical theory is to account for the way in which form and meaning are linked in natural languages, for example showing how phrases in a sentence pick out the participants and their roles (agent, patient, beneficiary etc.) in described situations. Within LFG, the link between syntactic phrases and participant (or argument) roles is mediated by an inventory of syntactic functions like SUBJ, OBJ, OBJth, OBL, XCOMP, etc.; this can be seen as a formalized version of well-known concepts from traditional grammar. There are cross-linguistic constraints on the way in which the arguments of a given predicate are mapped to syntactic functions in the f-structure, and there is a body of research that attempts to establish what these constraints are. The developing theory, Lexical Mapping Theory (LMT) [2, 6], has had to be continually revised as the number of typologically diverse languages investigated has increased. Tigrinya and Georgian raise different non-trivial problems for the application of LMT and hence provide a challenging and fruitful basis for approaching the questions of universal constraints on argument linking.

A multilingual parallel treebank provides information about the set of syntactic functions which is crosslinguistically available for a given argument. Our hypothesis is that LMT will allow the derivation of (possibly underspecified) information about the semantic role of an argument from such sets of alternative syntactic functions that can realize it. In a sense, semantic roles would then be labeled by their sets of alternative syntactic expressions in a way analogous to how alternative translations express the semantic properties of words in the Semantic Mirrors approach [5]. If the hypothesis is confirmed, this would be a highly interesting result both theoretically and practically. Our parallel treebank will provide a unique opportunity to approach these hypotheses, and similar ones, on a solid empirical basis.

3 Alignment principles

3.1 The intuitive notions

Parallel corpora are traditionally aligned on the sentence and word levels. On the sentence level, the default expectation is that all source sentences are aligned to one or (more rarely) more than one target sentence, and vice versa. Only complete sentence-formed omissions or additions in the translations lead to unaligned sentences.

On the word level, on the other hand, a source word is ideally aligned to a target word (or a target word sequence) if and only if they correspond translationally, where ‘translational correspondence’ in a text does not only mean semantic correspondence. Rather, words should be aligned only if their surroundings, too, correspond. ‘Surroundings’ in this connection must be defined in terms of the syntactically expressed argument structure of the sentence, within which the position of the word can be determined.

Thus, a source word W_S and a target word W_T are taken to correspond translationally only if (i) W_T can in general (out of context) be taken to be among the semantically plausible translations of W_S, i.e., W_T belongs to the set of ‘linguistically predictable translations’ (LPT) of W_S, and (ii) W_S and W_T occupy corresponding positions within corresponding argument structures. Henceforth we will refer to criterion (i) as ‘LPT-correspondence’ between a source and a target word. We assume that source/target nouns also enter into ‘LPT-correspondence’ with target/source pronominal forms. It remains to make precise the criteria for saying that a source argument structure and a target argument structure ‘correspond’; this is discussed in 3.2.

Phrase level correspondence is intermediate between sentence level correspondence and word level correspondence. The intuition behind the notion of translational correspondence on the phrase level is that a source phrase Ph_S and target phrase Ph_T are taken to correspond if (i) they contain corresponding words, (ii) Ph_S contains no word or phrase corresponding to a target word or phrase outside Ph_T, and similarly (iii) Ph_T contains no word or phrase corresponding to a source word or phrase outside Ph_S.

Given the criteria sketched above for word-level correspondence, we need not state as an independent criterion that corresponding phrases must express corresponding (in some sense) argument structures (or argument structure parts); they necessarily will if their words truly correspond. Furthermore, we do not require that Ph_S and Ph_T also occupy corresponding positions within translationally corresponding argument structures, as we assume on the level of word correspondences. Such a requirement might put too strong demands on the degree to which translationally corresponding sentences must also correspond syntactically in order for phrase and word alignment to occur. However, position within corresponding argument structures should be a criterion for ranking competing alternative phrase alignments.

This explication of the criteria for correspondence implies that there is a mutual dependence between correspondences on the different levels of a source/target sentence pair. Therefore we cannot first achieve full word-level alignment and only then go on to consider correspondences on the higher levels, nor can we do the opposite. Rather, we need a non-monotonic bootstrapping approach to alignment: based on a seed of an initial tentative, possibly partial and many-to-many word-level alignment relation, the alignment of argument structures takes place, which in turn may justify further word level alignments and also eliminate some of the initial alignments. We assume that this procedure will reach a fixed point. The linking of c-structure phrase nodes only takes place when such a fixed point has been reached and the word-level alignment is stable and one-to-one.

3.2 Phrase alignment based on parallel LFG analyses

In an LFG analysis, the argument structure properties of a phrase Ph are expressed in the value of PRED in the f-structure F which Ph 'projects' by the mapping function ϕ (see [4], ch. 4), in conjunction with other properties of F. The value of PRED is always a 'semantic form'. Let $L(Pr)$ be the lexical expression of the predicate Pr in a semantic form $Pr\langle ARG_1 \dots ARG_n\rangle$.[1] Furthermore, let $P(ARG_i)$ be the predicate in the semantic form of the f-structure to which ARG_i is argument-linked,[2] let $P(ADJ)$ be the predicate in the semantic form of an adjunct ADJ,[3] and let $F^{-\phi}$ be the set of c-structure nodes projecting the f-structure F.

A source f-structure F_S is said to 'correspond' to a target f-structure F_T if F_S and F_T have partially or fully corresponding PRED-values such that $PRED_S = Pr_S\langle ARG_{1S} \dots ARG_{nS}\rangle$ and $PRED_T = Pr_T\langle ARG_{1T} \dots ARG_{mT}\rangle$, where

(i) the number of arguments n and m may or may not differ,

(ii) there is LPT-correspondence between $L(Pr_S)$ and $L(Pr_T)$,

(iii) for each ARG_{iS}, there is LPT-correspondence between $L(P(ARG_{iS}))$ and either some $L(P(ARG_{j_T}))$ or some $L(P(ADJ_T))$ of an ADJ_T in F_T, and, conversely,

(iv) for each ARG_{iT}, there is LPT-correspondence between $L(P(ARG_{iT}))$ and either some $L(P(ARG_{j_S}))$ or some $L(P(ADJ_S))$ of an ADJ_S in F_S,

(v) the LPT-correspondences can be aligned one-to-one, and

(vi) there is no adjunct ADJ in F_S such that $L(P(ADJ))$ is word-aligned with a target node projecting an f-structure outside F_T, and vice versa for adjuncts in F_T.

This includes the special case when F_S and F_T have fully corresponding PRED-values $PRED_S = Pr_S\langle ARG_{1S} \dots ARG_{nS}\rangle$ and $PRED_T = Pr_T\langle ARG_{1T} \dots ARG_{nT}\rangle$, where

(i) the PRED-values have the same number of arguments $ARG_1 \dots ARG_n$,

(ii) there is LPT-correspondence between $L(Pr_S)$ and $L(Pr_T)$,

[1]Example: In the f-structure for the sentence *John sleeps*, the semantic form is 'sleep$\langle(\uparrow \text{SUBJ})\rangle$', and $L(\text{sleep})$ is the word form *sleeps*.

[2]Example: In the semantic form 'sleep$\langle(\uparrow \text{SUBJ})\rangle$' from the previous footnote, ARG_1 is argument-linked to the SUBJ, whose semantic form is 'John', which is hence the value of $P(ARG_1)$. The value of $L(P(ARG_1))$, then, is the word form *John*.

[3]Since the PRED-value of an adjunct may be supplied by a preposition, this definition must be sharpened to pick out the semantic form of the OBJ of the preposition in such cases. Thus, if ADJ is the f-structure of an adjunct *on the table*, $P(ADJ)$ would be the semantic form 'table'.

(iii) for every i, $1 \leq i \leq n$, there is LPT-correspondence between $L(P(ARG_{iS}))$ and $L(P(ARG_{iT}))$,

(iv) the LPT-correspondences can be aligned one-to-one, and

(v) there is no adjunct ADJ in F_S such that $L(P(ADJ))$ is word-aligned with a target node projecting an f-structure outside F_T, and vice versa for adjuncts in F_T.

Unlike this special case, the general case does not ensure meaning equivalence between the corresponding f-structures, since it allows corresponding arguments to occur in different orders in the semantic forms. This leaves the degree of semantic equivalence between translationally corresponding complex expressions to some extent open as an empirical question, and also exempts grammar writers from the requirement of achieving completely uniform cross-linguistic criteria for argument ordering, both of which freedoms we consider desirable.

In cases of null pronominal arguments, $L(P(ARG_i))$ is undefined, since there is no lexical expression of the argument. In such cases we assume that the requirement of LPT-correspondence is satisfied if there is a corresponding argument or adjunct in the other language (according to the other criteria above) which is either also a null pronominal argument or has a lexical expression which is not aligned with anything else.

In cases where a source f-structure corresponds to more than one target f-structure, or vice versa, the alternatives are ranked according to (i) the closeness of the correspondence (the special case above being closer than cases involving adjuncts, for example), and (ii) the occurrence vs. non-occurrence of the corresponding f-structures within corresponding embedding f-structures, where cases of corresponding embedding structures take priority.

Once corresponding f-structures have been identified according to the criteria above, all and only the word-alignment links which are in accordance with the f-structure correspondences are kept. In particular, this limits the alignment of nouns with pronouns (including null pronominals) to those cases which are motivated by the surrounding argument structures.

Now phrase alignment can be defined based on the correspondence relation between source and target f-structures and the concomitant word alignments. The set of nodes given by $F^{-\phi}$ constitutes a functional domain within the c-structure. All nodes within a functional domain are alignment candidates. However, we clearly cannot link all nodes in a source functional domain $F_S^{-\phi}$ to all nodes in the corresponding target functional domain $F_T^{-\phi}$. The reason is that as we move downwards in the functional domain along the head path in the c-structure, we may leave behind sister nodes contributing arguments and adjuncts to the shared f-structure (e.g. a subject NP as we move from the S mother to the VP daughter). But aligned nodes should only dominate corresponding material. Furthermore, in cases of long-distance dependencies there may be such contributing nodes that are not dominated within the functional domain at all. Hence, for a source node n_S, we need to make sure that we only align it with such target nodes n_T that dominate corresponding material (we do not align a source VP which does not dominate the subject NP with a target S dominating the translation of the source subject, even though the source VP and the target S project corresponding f-structures).

Given two corresponding f-structures F_S and F_T, this can be done by the following procedure. For every node n_S in $F_S^{-\phi}$ and every node n_T in $F_T^{-\phi}$, find the set $LL(n_S)$ of linked lexical nodes dominated by n_S (i.e., lexical nodes which are word-aligned with something in the target string), and find the set $LL(n_T)$ of linked lexical nodes dominated by n_T. Align n_S and n_T if and only if $LL(n_S)$ and $LL(n_T)$ are non-empty, all the nodes in $LL(n_S)$ are aligned with nodes in $LL(n_T)$, and vice versa.

Notice that these definitions leave open the possibility that the source or the target phrase may contain material, such as further adjuncts (but not further arguments), not corresponding to anything in the target or source, respectively. Given the frequency of additions and omissions in translations, we need that latitude.

4 The parallel treebanking tool

4.1 Functionality

To help our scientific exercise, we have built initial extensions of the LFG PARSEBANKER [10] to support parallel parsebanking. To our knowledge, there is no prior tool that adequately allows the creation, storage, visualization and search of translational correspondences at multiple levels of structure in parallel treebanks through a Web interface. We briefly describe its current functionality.

Since we currently do not have the bilingual resources required for the automatic performance of the initial tentative word alignment between our languages, the alignment of f-structures and words is presently done manually. The tool allows us to do this for corresponding sentences by dragging the index of a subsidiary source f-structure onto the index of the corresponding target f-structure. The alignment information is stored in a database as an additional layer. We envision doing this automatically in the future.

The procedure for the subsequent alignment of c-structure nodes presented in 3.2 is implemented in the tool, taking the manually aligned f-structures, with their concomitant word-alignments, as input.

4.2 Examples

We will illustrate by means of a few examples. In the screenshots in Figures 1–3 the Norwegian and Georgian subsidiary f-structures[4] have been aligned manually according to the criteria given in 3.2. The f-structure correspondences are shown in the indices on the substructures: an index of the form $\boxed{n\to m}$ tags structure

[4]The f-structures are shown in 'PREDs-only mode', i.e., many grammatical features, irrelevant to present purposes, have been suppressed.

n and indicates that it is aligned with structure *m*. The automatically derived c-structure alignments are shown by the curved lines. Nodes which share alignments are connected by heavy lines, and the alignment is marked only on the top member of such a set in order not to clutter up the representation unnecessarily. Dotted lines indicate distinct functional domains.

While Norwegian, like English, expresses the beneficiary either as an oblique prepositional phrase or an NP in a double object construction, Georgian only offers the latter possibility. A simple example of an alignment of the two constructions is provided in 1.

(1)	(a)	*Georg*	*ga*	*en*	*bok*	*til*	*Katarina.*
		George	gave	a	book	to	Catherine
		'George gave a book to Catherine.'					
	(b)	*gia-m*	*eḳa-s*	*ċign-i*	*misca.*		
		George-ERG	Catherine-DAT	book-NOM	he-gave-it-to_her		
		'George gave Catherine a book.'					

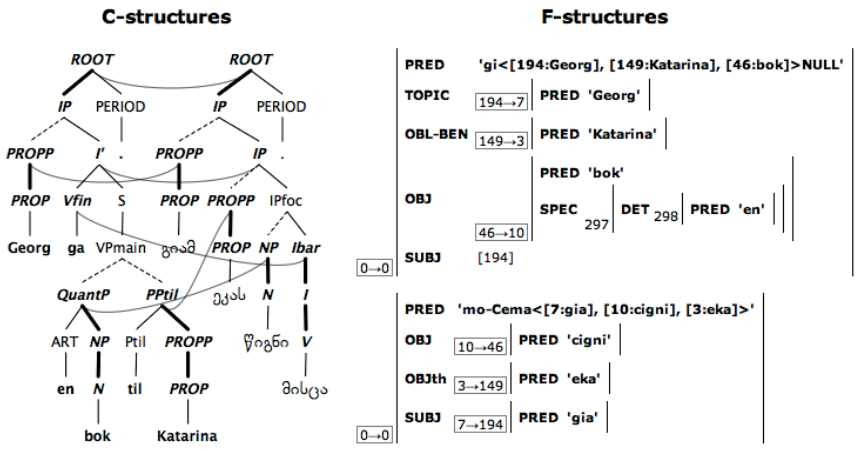


Figure 1: Screenshot of two-level alignment for Example 1

In Figure 1, the Norwegian SUBJ is aligned with the Georgian SUBJ (since the Norwegian SUBJ is unified with the TOPIC, the latter is also aligned with the Georgian SUBJ), the Norwegian OBJ is aligned with the Georgian OBJ, and the Norwegian OBL-BEN is aligned with the Georgian OBJth. As a consequence of the last-mentioned alignment the c-structure nodes PPtil and PROPP are aligned. Furthermore, we may notice that the two grammars happen to have the three arguments of *give* in different orders in the semantic forms, but this does not prevent alignment according to the criteria in 3.2.

Example 2 shows a case where an adjunct in Norwegian corresponds to an argument in Georgian.

(2) (a)

Også	*på*	*denne*	*konvolutt-en*	*stod*	*navn-et*	*hennes.*
Also	on	this	envelope-DEF	stood	name-DEF	her

‘Her name was on this envelope, too.’

(b)

am	*ḳonverṭ-sa-c*	*mis-i*	*saxel-i*	*eçera.*
this-DAT	envelope-DAT-too	her-NOM	name-NOM	it-was-written.

‘Her name was on this envelope, too.’

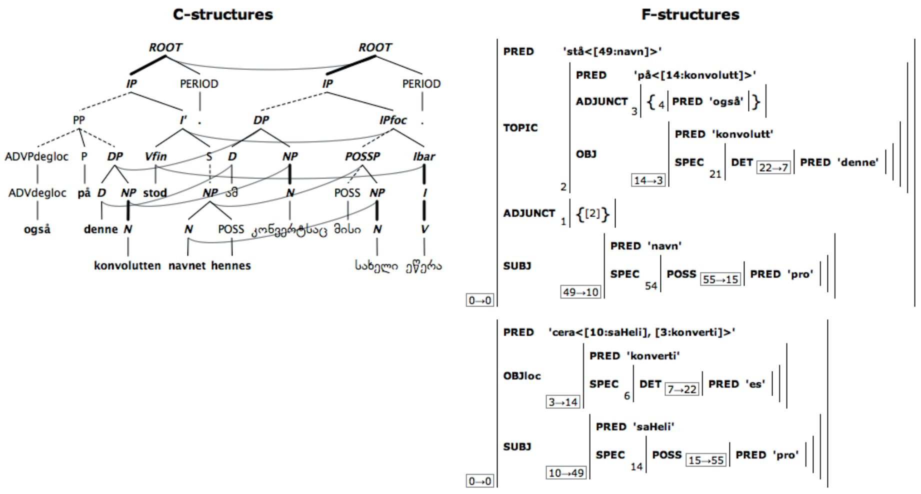


Figure 2: Screenshot of two-level alignment for Example 2

In the Norwegian f-structure in Figure 2, the TOPIC is identical with a member of ADJUNCT, and the OBJ of this shared value is aligned with OBJloc in the Georgian f-structure. As a result, the Norwegian DP under PP and the Georgian DP under IP are aligned.

A Norwegian-Georgian example involving a long-distance dependency (topicalization) in Norwegian, but not in Georgian, is shown in 3.

(3) (a)

Georg	*antar*	*jeg*	*du*	*mener.*
George	assume	I	you	mean.

‘George I assume you mean.’

(b)

vpikrob,	*rom*	*gias*	*gulisxmob.*
I-assume-it,	that	George-DAT	you-mean-him

‘I assume you mean George.’

In the Norwegian c-structure in Figure 3, the nodes ROOT, IP, PERIOD, I’, Vfin (the upper one), S and VPmain belong to the same functional domain, projecting the entire f-structure. Still, the nodes I’, S and VPmain do not enter into any

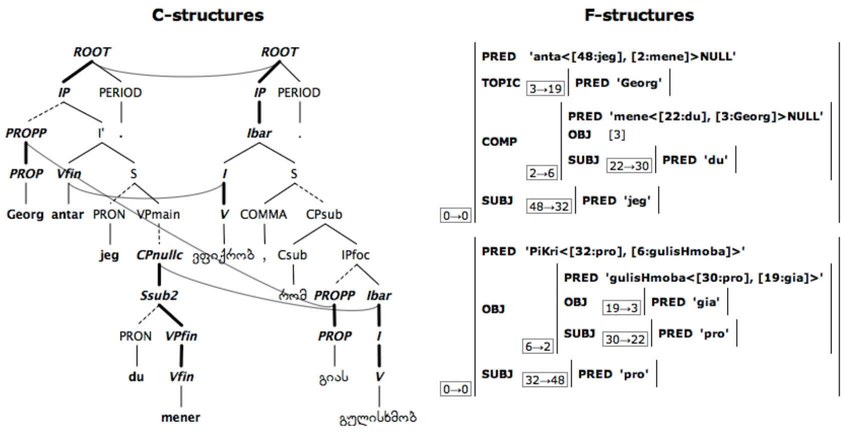


Figure 3: Screenshot of two-level alignment for Example 3

alignment because they do not dominate the word *Georg*, while the latter is word-aligned with a word dominated by the Georgian nodes ROOT, IP, Ibar (the upper one) and S, which would have been the alignment candidates.

On the other hand, the Norwegian nodes CPnullc, Ssub2, VPfin and Vfin (the lower one) are aligned with the Georgian nodes Ibar, I and V (the lower ones). The latter nodes dominate a verb (*gulisxmob*, 'you-mean-him') expressing the second person subject by inflection, while no overt pronoun in the sentence corresponds to the Norwegian second person pronoun *du*, which is therefore unaligned. Hence the criteria are satisfied for aligning both the Norwegian phrase *du mener* and the single verb *mener* with the phrase nodes dominating the single Georgian verb *gulisxmob*; see 3.2 on the treatment of lexically unexpressed PREDs like 'pro'.

However, the same Norwegian nodes CPnullc, Ssub2, VPfin and Vfin are *not* aligned with the Georgian nodes CPsub and IPfoc, although the latter belong to the same functional domain as the nodes Ibar, I and V below them. The reason is that CPsub and IPfoc dominate the name form *gias*, whose Norwegian word-alignment partner *Georg* is not dominated by CPnullc, Ssub2, VPfin or Vfin.

4.3 Search

The LFG Search tool [10] is being extended with a parallel search mode. For aligned sentence pairs, certain c-structure nodes and f-structure nodes (that is, subsidiary f-structures) will be aligned. To make alignment relations searchable, an alignment relation has been introduced as shown in Example 4.

(4) #s >>> #t

This relation holds if #s is instantiated by a node in the source c- or f-structure, #t is instantiated by a node in the target c- or f-structure, and those nodes are aligned.

Thus, the query in Example 5 will match all aligned pairs of analyses in a Norwegian-Dutch parallel treebank where a source c-structure lexical node *jente* is aligned with a target c-structure lexical node *meisje*.

(5) #s:"jente" >>> #t:"meisje"

An alignment relation can of course be part of a more complex query expression, as Example 6 illustrates. This query will find examples, like Example 2, where an argument is aligned with an adjunct, that is, aligned f-structures f1 (instantiating #f1) and f2, where a subsidiary argument f-structure s1 in f1 is aligned with a subsidiary adjunct f-structure s2 in f2: [5]

(6) #f1 >ARG #s1 & #f2 >(ADJ $) #s2
& #f1 >>> #f2 & #s1 >>> #s2

In a parallel treebank a single sentence in one language may correspond to multiple sentences in another. This can be handled on the overview Web page where manual alignment is implemented using drag and drop.

5 Conclusion

In this paper we have introduced the theoretical and methodological starting points for a linguistically motivated parallel treebanking approach that includes formal criteria for alignment. Rooting phrase alignment in correspondences at the level of predicate-argument structure within a parsebanking method which is both empirically founded and formally constrained offers a new approach to the study of the syntax-semantics interface across languages.

In the long run, this might open a new route to discovering language universals in this area, but currently we are only starting to explore this approach on a small number of typologically diverse languages. We have reported on the construction of a tool, a first prototype of which is operative and is being tested on test suites. Our work on alignment needs refinement and testing. We are also extending and testing the grammars in an integrated parsebanking approach and intend to move towards parsebanking of naturally occurring texts.

[5] '>ARG' is an abbreviation for '>(SUBJ | OBJ | ... | PREDLINK)', that is, the set of governable grammatical functions, '$' is the set-membership operator, and the expression '#f2 >(ADJ $) #s2' matches all pairs of f-structures f2, s2 where s2 is a member of the set constituting the value of ADJUNCT of f2.

References

[1] Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, Malden, MA, 2001.

[2] Joan Bresnan and Lioba Moshi. Object asymmetries in comparative Bantu syntax. *Linguistic Inquiry*, 21(2):147–185, 1990.

[3] Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The Parallel Grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan*, 2002.

[4] Mary Dalrymple. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, San Diego, CA, 2001.

[5] Helge Dyvik. Translations as a semantic knowledge source. In *Proceedings of The Second Baltic Conference on Human Language Technologies*, pages 27–38, Tallinn, 2005. Institute of Cybernetics at Tallinn University of Technology, Institute of the Estonian Language.

[6] Anna Kibort. Extending the applicability of Lexical Mapping Theory. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG ’07 Conference*, pages 250–270. CSLI Publications, Stanford, 2007.

[7] Paul Meurer. A computational grammar for Georgian. In *Proceedings of the Seventh International Tbilisi Symposium on Language, Logic and Computation*, Tbilisi, Georgia, 2007.

[8] Nazareth Amlesom Kifle. Differential object marking and topicality in Tigrinya. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG ’07 Conference*, pages 5–25. CSLI Publications, Stanford, 2007.

[9] Victoria Rosén, Koenraad De Smedt, Helge Dyvik, and Paul Meurer. TREPIL: Developing methods and tools for multilevel treebank construction. In Montserrat Civit, Sandra Kübler, and Ma. Antònia Martí, editors, *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 161–172, 2005.

[10] Victoria Rosén, Paul Meurer, and Koenraad De Smedt. LFG Parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord, and Koenraad De Smedt, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 127–133, Utrecht, 2009. LOT.

[11] Yvonne Samuelsson and Martin Volk. Automatic phrase alignment: Using statistical n-gram alignment for syntactic phrase alignment. In Koenraad De Smedt, Jan Hajič, and Sandra Kübler, editors, *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories*, pages 139–150, 2007.

Clausal Coordinate Ellipsis and its Varieties in Spoken German: A Study with the TüBa-D/S Treebank of the VERBMOBIL Corpus

Karin Harbusch
University of Koblenz-Landau
Computer Science Department
Koblenz, Germany
harbusch@uni-koblenz.de

Gerard Kempen
Max Planck Institute
for Psycholinguistics
Nijmegen, The Netherlands
gerard.kempen@mpi.nl

Abstract

Grammar rules for Clausal Coordinate Ellipsis (CCE) are based nearly exclusively on linguistic judgments (intuitions). For German, the extent to which grammar rules based on this type of empirical evidence generate all and only CCE structures that populate text corpora, has only been explored with the TIGER treebank of written newspaper text. How well these rules fit spoken German is unknown. In this paper, we study the applicability of judgment-based CCE rules to spontaneously spoken German by means of the TüBa-D/S treebank, which is based on dialogues for appointment scheduling and travel planning from the VERBMOBIL project. The judgment-based CCE rules are shown to hold nearly equally well for spoken as for written text: The proportion of deviations from the rules are virtually identical—less than 3% of the utterances/sentences that include a clausal coordination (compared to about 1% in the TIGER treebank). Moreover, the relative frequencies in VERBMOBIL of four main CCE types distinguished in the literature reveal a pattern that resembles the pattern observed in CGN2.0, the Corpus of Spoken Dutch.

1 Introduction

Coordinating conjunctions often license syntactic constituents to be elided from one conjunct if they have a (nearly) identical counterpart in another conjunct. Example (1), taken from the TüBa-D/S[1] treebank, exhibits "forward" elision of *viertel vor zwölf könnte* combined with "backward" elision of *abholen*. The presumed ellipsis sites are indicated by dots. At those sites,

[1] TüBa-D/S contains 38,228 sentences with about 380,000 word tokens (Stegmann et al. [16]) collected in the VERBMOBIL project (see Wahlster [21]). Here, a sentence refers to a complete dialogue turn by a speaker and thus often consists of several main clauses. The dialogue partners schedule appointments and set up traveling plans. In the following, we refer to the TüBa-D/S as the *"VERBMOBIL"* treebank in order to avoid confusion with the TüBa-D/Z treebank (Hinrichs et al. [7]) which is a corpus of German newspaper texts currently comprising about 22,000 sentences taken from the Wissenschafts-CD of "die tageszeitung"—henceforth called the "*TAZ*" treebank.

the elliptical conjuncts may be said to BORROW overtly mentioned counterparts from the parallel conjunct. The example also illustrates a particularity of spoken language, namely *underreduction* with respect to backward elision of *Sie* 'you' from the first conjunct. (Without this *Sie*, we would analyze the example as a coordination of NPs rather than as clausal coordination.)

(1) *Viertel vor zwölf könnte ich Sie ... oder ... mein Fahrer Sie abholen*
Quarter to twelve could I you or my chauffeur you up-pick
'Quarter to twelve, I could pick you up or my chauffeur could do so'

Grammar rules for CLAUSAL COORDINATE ELLIPSIS (CCE) are based nearly exclusively on linguistic judgments (intuitions). In Harbusch & Kempen [3], we investigated how well a set of judgment-based rules aiming to generate all CCE varieties in German is obeyed in written language. For the TIGER treebank of newspaper texts (Brants et al. [1]), we reported 99% accuracy. How well spoken language fits these rules is not known. In the following, we present a qualitative and quantitative study of CCE in the VERBMOBIL (TüBa-D/S) treebank of spoken German dialogues. To anticipate one of the main results, the judgment-based CCE rules accord with more than 97% of the CCE tokens. However, the error types overlap only partly with CCE errors in written German.

The paper is organized as follows. In Section 2, we present an overview of the main types of CCE and spell out the rule set proposed by Kempen [9] and Harbusch & Kempen [5]. In Section 3, we briefly describe corpus studies on coordinate structures in German, Dutch and English reported in the literature. In Section 4, we present our study on CCE in the VERBMOBIL treebank. Moreover, we compare the German results to English and Dutch findings. Finally, in Section 5 we draw some conclusions and mention a desideratum for future work.

2 The Main CCE Types and How to Generate Them

In the linguistic literature on coordinate syntactic structures (for overviews, see van Oirsow [19]; Steedman [14]; Sag et al. [14]; te Velde [17]; and Kempen [9]), one often distinguishes four main types of coordinate ellipsis:[2]

- GAPPING, with three special variants called LONG DISTANCE GAPPING (LDG), SUBGAPPING, and STRIPPING,
- FORWARD CONJUNCTION REDUCTION (FCR),
- BACKWARD CONJUNCTION REDUCTION (BCR; also known as *Right Node Raising* or *RNR*), and
- SUBJECT GAP WITH FINITE/FRONTED VERB (SGF).

[2]We do not deal here with the elliptical constructions known as VP Ellipsis, VP Anaphora and Pseudogapping because they involve the generation of pro-forms instead of, or in addition to, the ellipsis proper. For example, *John laughed, and Mary did, too*—a case of VP Ellipsis—, includes the pro-form *did*. Nor do we account for recasts of clausal coordinations as coordinate NPs (e.g., changing *John likes skating and Peter likes skiing* into *John and Peter like skating and skiing, respectively*). Presumably, such conversions involve a logical rather than syntactic mechanism.

Table 1. Clausal coordinate ellipsis (CCE) types as specified by Harbusch & Kempen [5]. Column 1 mentions the names of the CCE types and in brackets their abbreviations. Column 2 illustrates the CCE types in terms of English examples. The distinctions apply to German as well. Column 3 summarizes the elision conditions we apply in our study. Struck-out text represents borrowings.

CCE type	**Examples**	**Elision conditions**
Gapping (g)	(2) *Ulf* ***lives*** *in Leipzig and* *his children* ***~~live~~***$_g$ *in Ulm*	Lemma identity of Verb & contrast of remnants
LDG ((g)$^+$g)	(3) *My wife* ***wants to buy*** *a car and* *my son* ***~~wants~~***$_g$ ***[~~to buy~~]***$_{gg}$ *a motorcycle*	Gapping conditions in a *superclause*
Sub-gapping (sg)	(4) *The driver* ***was*** *killed and* *the passengers* ***~~were~~***$_{sg}$ *severely wounded*	Gapping conditions & VP remnant in second conjunct
Stripping (str)	(5) *Mare* ***lives in Narva*** *and* *her children* ***[~~live in Narva~~]***$_{str}$ ***too***	Gapping conditions & only one non-Verb remnant
FCR (f)	(6) ***Since a year, Kees*** *lives in Aam and* ***[~~since a year, Kees~~]***$_f$ *works in Edam* (7) *Tokyo is the city [**where** Ota lives and* ***~~where~~***$_f$ *Kusuke works]*$_S$	Wordform identity & left-peripherality (within clause boundaries) of major clausal constituents
BCR (b)	(8) *John wrote one* ***~~article~~***$_b$ *and* *Mary edited two* ***articles****.* (9) *Anja arrived before three* ***[~~o'clock~~]***$_b$ *and* *Maria* ***~~arrived~~***$_g$ *after four* ***o'clock***	Lemma identity & right-peripherality, possibly disregarding major constituent boundaries
SGF (s)	(10) *Into the wood went* ***the hunter*** *and* ***[~~the hunter~~]***$_s$ *shot a hare*	Form-identical Subject & first conjunct starting with Verb/Modifier/Adjunct & FCR applied if licensed

As summarized in column 3 of Table 1, all forms of *Gapping* are characterized by elision of the posterior member of a pair of lemma-identical Verbs. The position of this Verb need not be peripheral but is often medial, as in (2) through (5), and (9).[3] Every non-elided constituent (remnant) in the posterior conjunct should pair up with a constituent in the anterior conjunct that has the same grammatical function but is not coreferential.[4] Stated differently, the members of such a pair are CONTRASTIVE—in (2): the Subjects *Ulf* vs. *his children*, and the locative Modifiers *in Leipzig* vs. *in Ulm*. Notice that although the two tokens of *my* in (3) occupy comparable positions in the two conjuncts, it is not possible to elide one of them because all CCE construc-

[3] In our definitions of CCE types, we restrict ourselves to coordinations encompassing two conjuncts, called *anterior* (first, left) and *posterior* (second, right), respectively.

[4] In the following, we distinguish three identity relationships between constituents in coordinated conjuncts: lemma identity, wordform identity and coreferentiality. For *lemma identity,* only the lexical entries ('syntactic words') of the constituents have to be identical; *wordform identity* requires, in addition, identity of their morphological features. *Coreferential constituents* refer to the same discourse entity or entities, irrespective of whether or not they include the same lemmata.

tions except BCR respect major constituent boundaries. On the other hand, *were* in (4) can be elided from the posterior conjunct although it has no wordform-identical (but only a lemma-identical) anterior counterpart.

In LDG, the remnants originate from different clauses (more precisely: from different clauses that belong to the same SUPERCLAUSE; a superclause is a hierarchy of finite or nonfinite clauses that do not include a subordinating Conjunction—with the possible exception of the topmost clause). In (3), *my son* belongs to the main clause but *a motorcycle* to the infinitival complement clause. In SUBGAPPING, the posterior conjunct includes a remnant in the form of a nonfinite complement clause (VP; *severely wounded* in (4)). In STRIPPING, the posterior conjunct is left with one non-Verb remnant, often supplemented by a sentential Adverb such as *too* or *not.*

In FCR, elision affects the posterior token of a pair of left-peripheral strings consisting of one or more wordform-identical major constituents. In (6), the posterior tokens of *since a year, Kees* and *where,* respectively, belong to such pairs and are eligible for FCR.

BCR is almost the mirror image of FCR as it deletes the anterior member of a pair of right-peripheral lemma-identical word strings (*o'clock* in (9)); however, BCR may elide part of a major constituent—e.g. only the part *article* of the Direct Object in (8) and *o'clock* of the temporal Modifier *before three o'clock* in (9). In addition, it requires only lemma identity (cf. example (8)).

SGF can elide the Subject of the posterior conjunct—always a main clause—when in the anterior conjunct the wordform-identical Subject follows the finite Verb (Subject-Verb inversion). Elision of the posterior Subject cannot be due to FCR since the anterior Subject is not left-peripheral. Furthermore, the initial constituent of an anterior SGF conjunct should NOT be an argument. This is illustrated by the ill-formed ellipsis in example (11) where a Complement clause opens the anterior conjunct. (In well-formed SGF case (10), the initial constituent is an Adjunct.)[5]

(11) **Das Examen bestehen will er und ~~er~~$_s$ kann auch*
The exam to-pass wants he and can too
'He wants to pass the exam and will be able to as well'

3 Previous Corpus Work on CCE in German, Dutch and English

In a recent paper (Harbusch & Kempen [3]), we analyzed the incidence of clausal coordination and coordinate ellipsis in the TIGER treebank. TIGER contains 50,474 syntactically annotated sentences originating from a German newspaper corpus. Almost 43% of them (21,506 sentences include a coordinate structure of any type, and 7,194 sentences (33% of the latter) contain at least one clausal coordination. In total, 4,020 TIGER sentences contain at

[5]We also subsume under the heading of SGF cases like (i), where the anterior conjunct is a conditional subordinate clause. See Höhle [8] and Reich [13] for discussion of the affinity between this structure and SGF as defined here.

(i) *ja, dann reicht es ja, wenn wir ungefähr um neun losfahren würden und ~~wir~~ würden dann mittags dort ankommen*
'OK, then it suffices if we would leave at nine and would arrive there in the afternoon'

least one CCE token, distributed over the four main CCE types as follows: 2545 cases of FCR (63%), 678 Gapping tokens (17%), 384 SGF cases (10%), and 413 BCR tokens (10%).[6] Of these, 99% percent obey the rules of Table 1. Only some 40 sentences violate a borrowing rule but were judged at least marginally acceptable. These sentences embody four borrowing (elision) patterns that may be characterized as 'fringe deviations' from the intuition-based coordinate ellipsis rules: *overreduction, peripherality violations by little words, peripherality violations by content words or word groups*, and *sloppy gapping*.

For Dutch, we conducted a comparative study of CCE in written and spoken language ([4][6]). We explored the ALPINO treebank (van der Beek et al. [18]) consisting of 7,153 manually annotated syntactic structures from a newspaper corpus, and CGN2.0 (van Eerten [19]) with about 130,000 spoken sentences or dialogue turns from more than ten different domains. In written Dutch, the percentage of elliptical versions within the set of all clausal coordinations is three times higher than in spoken Dutch: 34% versus 11% (Harbusch & Kempen [4]). In each of the treebanks, Gapping and FCR together covers 92% of the CCE cases (with the remaining 8% more or less evenly distributed among SGF and BCR). However, the distribution of FCR and Gapping in the two Dutch treebanks differs widely. Whereas in written clausal coordinations Gapping accounts for only 10% of the CCE cases (with a large majority of 82% embodying FCR), in spoken clausal coordinations the incidence of Gapping is much higher: 31% (leaving 61% for FCR). These numbers are comparable to those observed in the German written and spoken corpora (the latter are reported in the next Section).

In two corpus studies into the incidence of CCE in spoken and written English, Meyer [12] and Greenbaum & Nelson [2] found that in written clausal coordinations, the proportion of elliptical versions is about twice as high as in spoken coordinations.

These findings suggest that there may be substantial cross-linguistic similarities, at least as far as the Germanic languages are concerned, with respect to the frequencies of CCE and CCE types in spoken texts and in written texts.

4 Clausal Coordinate Ellipsis in the VERBMOBIL Treebank

After an outline of the methodology of the corpus study, we report on the accuracy of the elision rule set and the error classes found in the VERBMOBIL corpus. Finally, we compare our frequency results to the findings for Dutch and English reported in the previous Section.

[6]In a quantitative study into the German TAZ treebank, Zinsmeister [22] found 8,133 sentences (37% of the total number of sentences) that include a coordination of syntactic constituents of any type (marked by the edge label KONJ). She only reports one number dealing with CCE types: 83 sentences with SGF—i.e about 1% of all coordinations. This percentage is comparable to the proportion of SGF cases in TIGER: the 384 cases we observed there, make up less than 2% of the total number of coordinations.

4.1 Methodological Issues

The VERBMOBIL treebank is encoded in the same manner as the TAZ corpus (Hinrichs et al. [7]) but rather differently from TIGER (see Lemnitzer & Zinsmeister [11], page 82, for a comparison of the tag sets).

In TIGER, coordinate elisions are explicitly marked by SECONDARY EDGES, i.e. edges that run from the root node of a remnant—a borrowed string—to the root node of the structure that borrows the remnant as a child node. The edge label indicates the grammatical function that the borrowed remnant fulfils in the borrowing structure. These encodings enable the composition of search queries that automatically retrieve CCE structures (by means of TIGERSearch; König & Lezius [10]). Moreover, they support semi-automatic classification of CCE types and verification of the elision rules.

Secondary edges do not occur in VERBMOBIL trees. Therefore, we manually inspected all clausal coordinations for CCE, classified them for CCE type, and marked all rule violations. Additionally, we accessed and checked all dialogue turns from which the CCE tokens originate and ruled out any false alarms generated by the search queries. This procedure enabled us to compare the frequency data for written text in TIGER with the frequencies of spoken text in VERBMOBIL.

The VERBMOBIL frequency counts proceeded in two steps. First, we collected all clausal coordinations consisting of one or more incomplete conjuncts—incomplete in the sense that some constituent(s) seemed to be missing. Utterances that we judged to be ill-formed due to a self-correction by the speaker, like sentence (12), were left out of consideration. This also happened to over 100 cases which include a left-dislocated constituent followed by a resumptive pro-form—*das* 'that' in (13)—, which one could tentatively analyze as opening the posterior conjunct of an asyndetic coordination, with deletion of a right-peripheral string in the first conjunct, as in BCR.

(12)	*da*	*hat*	*da*	*hätte*	*ich*	*auch*	*Zeit*
	then	have	then	would-have	I	too	time

'then I would have time as well'

(13)	*aber*	*siebzehnter,*	*achtzehnter*	~~*ginge*~~	*das*	*ginge .*
	'but	seventeenth,	eighteenth		that	would-be-possible'

We also ruled out all cases, which we judged to result from plausible *conceptual inference* rather than from borrowing licensed by a coordinating conjunction. In (14), the Adverb 'then' probably modifies not only the anterior but also the posterior conjunct. However, the absence of does not render the second conjunct incomplete. Hence, we classified (14) as a well-formed case of FCR with borrowing of the Personal Pronoun 'we' only. (For details regarding conceptual inference, see Harbusch & Kempen [3].)

(14)	*wir*	*fliegen*	*dann*	*am*	*elften*	*und*	~~*wir*~~	*bleiben*	*für*	*zwei*	*Tage*
	,'we	fly	then	on-the	eleventh	and		stay	for	two	days'

Importantly, we only considered structures that do not allow an alternative analysis as a nonclausal coordination of NPs, PPs, APs, etc. For instance, sentences (15) and (16) were discarded due to the possibility of analyzing them as PP-coordination (instead of as a combination of BCR and FCR—cf. example (1)). Importantly, however, just as in our TIGER study, we included nonclausal coordinations into the CCE counts if the posterior conjunct follows the clause-final Particle or Verb of the anterior conjunct—see (17) and (18) for an illustration. In the classification of CCE types, we group them together with the Stripping variant of Gapping.

To prevent a misunderstanding, whenever a VERBMOBIL utterance of the type discussed here had been encoded explicitly either as a discontinuous structure or as Stripping/Gapping, we adopted this choice. (In (17), PP *nach Hannover* was encoded as an extraposed part of the NP headed by *Reise*; and *und zwar* was encoded as a discourse marker rather than as a syntactic node.) However, very often the encodings left the choice between discontinuous structure vs. Stripping/Gapping open.

(15) *dann sage ich meiner Sekretärin [wegen der Bahnkarten und wegen dem Hotel]$_{PP}$ Bescheid*
‘Then I’ll inform my secretary about the tickets and about the hotel’

(16) *ich war schon ein paar Mal [in Hannover und zwar in dem Hotel Loccumer-Hof]$_{PP}$*
‘I was already a few times in Hannover, namely in hotel Loccumer-Hof’

(17) *ich habe eine Reise vor, und zwar nach Hannover*
I have a trip in-mind namely to Hannover

(18) *schauen wir noch, ob wir noch ins Theater gehen oder in ein Kino*
look we also whether we also to-the theater go or to a cinema
‘let’s also look whether we go to the theater or to a cinema’

In the second step, we classified the CCE tokens according to CCE type. Like in our TIGER study, when a sentence embodies several CCE constructions, we counted each of them separately. Recall that, in VERBMOBIL, sentence numbers were assigned to entire dialogue turns, which often include several (main) clauses. Sentence (19), for example, features two FCR cases, actually borrowing different left-peripheral strings.

(19) *wenn ich da nicht da wäre und ~~wenn~~$_f$ er in meinem Büro sitzen würde und ~~wenn er~~$_f$ Däumchen drehen würde*
‘if I wouldn’t be in and he would sit in my office and kick his heels’

When a CCE instance could be ranged under more than one type, we followed the encodings the TIGER treebank as much as possible. For cases like (20), for instance, we chose the FCR analysis although the sentences can be viewed as nonelliptical coordinations of infinitival clauses.

(20) *Oder möchten Sie sparen und ~~möchten~~ ~~Sie~~ das Doppelzimmer nehmen?*
‘Or would you like to save money and take a double room?’

Finally, while carrying out these steps, we sometimes needed to ‘clean up’ the sentence materials, for instance, to remove interjections or to insert words that were missing for reasons clearly unrelated to coordinate ellipsis. In (21), the Subject NP *Sie* ‘you’ seems to be missing after the second token of *wenn* ‘if’. (Given this reconstruction, the sentence is analyzed as a Stripping variant of Gapping.) In (22), the speaker interrupts the PP headed by *zwischen* ‘between', inserts a series of editing terms and interjections (the string between vertical bars), and resumes with *vielleicht* 'perhaps' and a revised PP. We disregarded the string between bars and interpreted the sentence as Gapping, with a contrast between *sehr gut* 'very well' and *vielleicht* on the one hand, and between the original and the revised dates on the other.

(21) *wenn Sie möchten, wenn (Sie) sich vielleicht das Museum-für-Hamburgische-Geschichte ansehen oder ~~wenn~~ (~~Sie~~) ~~sich~~ die Kunsthalle, die neu eröffnet worden ist ~~ansehen~~*
'If you like, if (you) visit the historical museum of Hamburg or the art gallery which has just reopened'

(22) *sehr gut passen würde es mir zwischen siebten Mai und || nee, ach doch das ist doch nicht gut das ist gar || vielleicht ~~passen~~ ~~würde~~ ~~es~~ ~~mir~~ zwischen dem achten Juni und elften Juni*
'very well would suit me between the 7th of May and || no, oh yeah, this is not good, this is even || perhaps between the 8th and 10th of June'

4.2 CCE Error Types in the VERBMOBIL Treebank

We found 3,713 VERBMOBIL sentences (or rather dialogue turns) with at least one clausal coordination (including asyndetic ones). This set includes 1,314 sentences (35%) with at least one CCE token. (See the next Subsection for the frequencies of the individual CCE types.) In the remainder of this Section, we concentrate on two questions: How well does spoken language obey the elision rules of Section 2, compared to written language? And to what extent do the error types observed in spoken language overlap with those seen in written language?

In VERBMOBIL, we identified 35 CCE tokens that violated a judgment-based CCE rule. That is, less than 3% of the sentences was ill-formed—a proportion not substantially different form the 1% deviations we reported earlier for written German. We find this somewhat surprising, given that spoken language is supposed to be more error-prone than written language: The speaker is under higher time pressure, has no external memory, and cannot easily hide away editing actions from the audience. On the other hand, the conceptual and grammatical structure of spoken sentences tends to be much simpler than that of written sentences (cf. average sentence length).

Of the 35 CCE error tokens, only a minority could be ranged unequivocally under the error classes we distinguished in Harbusch & Kempen [3]. We classified 13 tokens as OVERREDUCTIONS. In errors of this type, the elision process cuts into a major clause constituent functioning as remnant, with the consequence that only part of the remnant survives. In Stripping/Gapping example (23), the speaker failed to repeat *Fahrschein* 'ticket' in the posterior conjunct (in addition to deleting the Prepositon *aus* 'leaving' at the end of the PP). In (24), also classified as Stripping/Gapping, the Preposition+Article *ins* was left out in front of *Schauspiel* 'theater play'.

(23) *... ob es möglich ist einen Fahrschein von Dammtor aus zu*
whether it possible is a ticket from Dammtor leaving to
bekommen und nicht vom Hauptbahnhof
get and not from main-station
'... whether it is possible to get a ticket leaving from Dammtor and not one leaving from main station'

(24) *vielleicht könnten wir ins Theater gehen, Schauspiel, oder in eine Oper*
maybe can we to-the theater go play or to an opera
'maybe we can go to the theater, the playhouse or the opera'

Another error class we had dubbed SLOPPY GAPPING: The verb elided from the posterior conjunct is used with a subcategorization frame different from that of its counterpart in the anterior conjunct. For instance, the Verb *werden* 'be' is used as passive Auxiliary in one clausal conjunct and as a copula Verb in another. In VERBMOBIL, we found 4 coordinations that arguably feature this type of error. In (25), *mögen* 'like' functions first as modal Verb, then as transitive Verb. In the anterior conjunct of (26), *mögen* is used as intransitive Verb; the posterior conjunct requires the modal Verb *mögen.*

(25) *weiß nicht, in die Oper möchte ich grade nicht gehen, oder ins Konzert, aber vielleicht irgendwas kleines gemütliches Treffen*
'don't know, I wouldn't like to go to the opera or to a concert, but perhaps something small, (a) cosy meeting'

(26) *und ich möchte dann schon am fünfzehnten noch mal schnell ins Büro und schauen was sich da so angesammelt hat*
'and then already on the 15th I'd like (to go) quickly into the office and look what has been piling up there'

In both overreduction and sloppy gapping, the speaker presumably does not accurately take into account the constraints imposed by the syntactic shape of the anterior conjunct.

Of the two remaining error classes (peripherality violations by little words, or by content words or word groups), we could not find a single unequivocal instance in VERBMOBIL. However, in 18 dialogue turns we spotted a mixed bag of other imperfections. In (27), for instance, the Particle *zurück* 'back' does not have a contrastive counterpart; prefixing *fahren* with Particle *hin* 'away' would have made for a perfect Gapping structure. In (28), an FCR case gone awry, the right conjunct needs an initial adverbial modifier like *da* 'there' but the left conjunct only has the Subject NP *das* 'that' on offer.

(27) *ich würde also gern am am Montag vormittags fahren und*
I would thus gladly on-the on-the Monday morning travel and
am Freitag nachmittags zurück ...
on-the Friday afternoon back
'I'd like to travel Monday morning and (come) back Friday afternoon'

(28) *das ist direkt am Hauptbahnhof und kostet das Einzelzimmer*
that is directly at-the main-station and costs the single-room
einhundert und neunundzwanzig Mark.
one-hundred and twenty-nine Mark
'that is directly at the main station and (there) a single room is 129 DM'

Quite a few dialogue turns consist mainly of utterances in telegram style. We classified some 25 exemplars as clausal coordinations if at least one of the conjuncts includes a finite verb. In the majority of those coordinations (some asyndetic), the clause-initial topic position of both conjuncts is empty: ‘topic drop’; cf. (29) and (30). As the fillers of both topic positions are identical, we provisionally assume that the anterior filler is reconstructed from context, and that the posterior filler is borrowed from the—then reconstructed—anterior filler. If correct, this entails that we need not classify these cases as CCE errors but as legal CCEs.

(29) *startet Bonn Hauptbahnhof um acht Uhr fünfundvierzig und kommt an*
leaves Bonn main-station at 8 hour 45 and arrives
in Hannover Hauptbahnhof um zwölf Uhr vier
in Hannover main-station at 12 hour 4

(30) *liegt zentral, hat Hallenbad und Fitneßraum*
is located centrally has indoor-pool and fitness-room

4.3 CCE Frequencies in the VERBMOBIL Treebank, and Comparison with Other Studies

At the end of Section 3, we hypothesized that the CCE frequencies in spoken and written German, Dutch and English texts would exhibit similar patterns. The evidence obtained from the VERBMOBIL corpus supports this hypothesis. In all three languages, the proportion of CCE sentences within the total set of coordinated clauses is substantially higher in the spoken than in the written modality.

Table 2 shows the relative frequencies in German and Dutch of the four CCE types we distinguish. It reveals striking within-modality and within-language similarities. In the spoken modality, the incidence of Gapping is higher than in written language, mainly at the expense of FCR. In the German treebanks, BCR and SGF are well represented (in particular SGF) whereas in the Dutch corpora they live a somewhat marginal existence.

Table 2. Relative frequencies of the four types of CCE, expressed as percentages of the total set of sentences exhibiting CCE.

CCE type	Spoken language		Written language	
	VERBMOBIL (German)	**CGN 2.0 (Dutch)**	**TIGER (German)**	**ALPINO (Dutch)**
GAPPING	33	31	17	10
FCR	55	61	63	82
BCR	1	3	10	5
SGF	11	5	10	3

5 Discussion

We investigated to which extent Grammar rules for Clausal Coordinate Ellipsis, which are nearly exclusively based on linguistic judgments (intuitions), hold for spoken German. We followed the lead of a similar study conducted recently with the TIGER treebank for written German, using the TüBa-D/S treebank, which is based on dialogues for appointment scheduling and travel planning from the VERBMOBIL project. After having presented a set of judgment-based CCE rules for four main CCE types distinguished in the literature, we showed that these rules fit spoken text nearly equally accurately as written text: The proportions of utterances deviating from the rules are very similar—less than 3% of the spoken sentences that include a clausal coordination, compared to about 1% of the written sentences in the TIGER treebank. However, the rule violations in the spoken corpus turned out to be of a rather different nature than those in the written corpus. Furthermore, we found that the relative frequencies in VERBMOBIL of the four main CCE types reveal a pattern that strongly resembles the patterns observed in the CGN2.0 treebank, the Corpus of Spoken Dutch.

We conclude not only that parsers and generators for spoken German can rely on the intuition-based rule systems for CCE, in particular rules such as described in Section 2 above, but also that their performance can profit from measures that allow for the fringe deviations observed in Section 4.

In future work we hope to provide a psycholinguistic explanation for the frequency/error patterns obtained in the present study and its predecessors.

References

[1] Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G. and Uszkoreit, H. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation, 2,* 597-620.

[2] Greenbaum, S. and Nelson, G. (1999). Elliptical clauses in spoken and written English. In: Collins, P. and Lee, D. (Eds.). *The clause in English.* Amsterdam: Benjamins.

[3] Harbusch, K. and Kempen, G. (2007). Clausal coordinate ellipsis in German: The TIGER treebank as a source of evidence. In: *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, Tartu, Estonia.

[4] Harbusch, K. and Kempen, G. (2009a). A treebank study of clausal coordinate ellipsis in spoken and written language. In: *Proceedings of the 15th Annual Conference on Architectures and Mechanisms of Language Processing (AMLaP2009)*, Barcelona, Spain.

[5] Harbusch, K. and Kempen, G. (2009b). Generating clausal coordinate ellipsis multilingually: A uniform approach based on postediting. In: *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, Athens, Greece.

[6] Harbusch, K. and Kempen, G. (2009c). Incremental sentence production inhibits clausal coordinate ellipsis: A comparison of spoken and written language. In: *Proceedings of the Workshop on Incrementality in Verbal Interaction*, Bielefeld.

[7] Hinrichs, E., Kübler, S., Naumann, K., Telljohann, H. and Trushkina, J. (2004). Recent Developments in Linguistic Annotation of the TüBa-D/Z treebank. In: *Proceedings of the Third Workshop on Treebanks and Linguistic Theories* (*TLT04),* Tübingen.

[8] Höhle, T.N. (1990). Assumption about asymmetric coordination in German. In: Mascaró, J. and Nespor, M. (Eds.), *Grammar in Progress: Glow Essays for Henk van Riemsdijk*, 221–235. Dordrecht: Foris.

[9] Kempen, G. (2009). Clausal coordination and coordinate ellipsis in a model of the speaker. *Linguistics, 47,* 653-696.

[10] König, E. and Lezius, W. (2003). *The TIGER language: A Description Language for Syntax Graphs, Formal Definition*. Tech. Rep. IMS, University of Stuttgart.

[11] Lemnitzer, L. and Zinsmeister, H. (2006). *Korpuslinguistik: Eine Einführung.* Tübingen: Narr Studienbücher.

[12] Meyer, C.F. (1995). Coordination Ellipsis in Spoken and Written American English. *Language Sciences, 17*, 241-169.

[13] Reich, I. (2008). From discourse to ‘‘odd coordinations’’: On asymmetric coordination and subject gaps in German. In: Fabricius-Hansen, C. and Ramm, W. (Eds.), *‘Subordination’ versus ‘coordination’ in sentence and text: A cross-linguistic perspective.* Amsterdam: Benjamins.

[14] Sag, I.A., Wasow, T. and Bender, E.M. (2003). *Syntactic Theory: A formal introduction*. Stanford CA: CSLI publications [Second Edition.]

[15] Steedman, M. (2000). *The syntactic process.* Cambridge MA: MIT Press.

[16] Stegmann, R., Telljohann, H. and Hinrichs, E. (2000). S*tylebook for the German Treebank in Verbmobil*. Saarbrücken: DFKI Rep. 239.

[17] te Velde, J.R. (2006). *Deriving Coordinate Symmetries*. Amsterdam: Benjamins.

[18] van der Beek, L., Bouma, G., Malouf, R. and van Noord, G.-J. (2002). The Alpino Dependency Treebank. In: *Computational Linguistics in the Netherlands CLIN 2001*. Amsterdam: Rodopi.

[19] van Eerten, L. (2007). Over het Corpus Gesproken Nederlands. *Ne–derlandse Taalkunde*, *12*, *3*, 194-215.

[20] van Oirsow, R.R. (1987). *The syntax of coordination*. London: Croom Helm.

[21] Wahlster, W. (Ed.) (2000). *Verbmobil: Foundations of Speech-to-Speech Translation.* Berlin: Springer.

[22] Zinsmeister, H. (2006). Treebank Data as Linguistic Evidence? Coordination in TüBa-D/Z. *Pre-Proceedings of the International Conference on Linguistic Evidence,* Tübingen.

Dependency Annotation of Wikipedia: First Steps Towards a Finnish Treebank

Katri Haverinen,[1,3] Filip Ginter,[1] Veronika Laippala,[2]
Timo Viljanen,[1] Tapio Salakoski[1,3]

[1]Department of Information Technology,
[2]Department of French studies
[3]Turku Centre for Computer Science (TUCS)
20014 University of Turku, Finland
`first.last@utu.fi`

Abstract

In this work, we present the first results obtained during the annotation of a general Finnish treebank in the Stanford Dependency scheme. We find that the scheme is a suitable syntax representation for Finnish, with only minor modifications needed. The treebank is based on text from the Finnish Wikipedia, ensuring its free distribution and broad topical variance. To assess the suitability of Wikipedia text as the basis of a treebank, we analyze its grammaticality and find the quality of the language surprisingly high, with 97.2% of the sentences judged as grammatical. The treebank currently consists of 60 fully annotated articles and is freely available.

1 Introduction

Treebanks are among the most crucial resources for the development of natural language processing (NLP) methods. There exist a number of national treebanks for a variety of languages, including widely used and studied ones, such as English, as well as languages spoken by comparatively smaller populations, for example Slovene. For Finnish, no such treebank currently exists, considerably restricting the possibilities for NLP research for this language. To address this obvious deficiency, we have commenced an effort to develop the first Finnish language treebank and, in this paper, present the first results of this project.

The source of text for the treebank is the Finnish Wikipedia. One of its major advantages is that it is released under a free license, enabling the distribution of the resulting treebank at no cost and with no copyright issues. Apart from offering a great topical variety, the text is written collaboratively by a number of authors and thus also reflects a number of different personal writing styles. Since there is little prior work on Wikipedia-based treebanking, we assess the grammaticality of the language and thus, to some extent, its suitability for a source of treebank text.

The annotation scheme of the treebank is the well-known Stanford Dependency (SD) scheme which was designed specifically for NLP applications [1, 10]. The Finnish treebank is the first general language corpus annotated natively in the SD scheme. Since the scheme was originally designed for English, we discuss its applicability to Finnish as part of the results presented in this paper. In particular, we show that only minor modifications to the scheme are necessary. The choice of the scheme follows a recent substantial interest in the application of dependency schemes in general and the numerous successful applications of the SD scheme specifically [8, 10, 12].

Among the most important application areas for treebanks is the induction and evaluation of statistical parsers. For instance, a number of national treebanks for diverse languages such as Catalan, English, and Japanese have been used in the recent CoNLL'09 shared task [2] to develop and evaluate multilingual statistical parsers, thus greatly benefiting the NLP research for these languages. Indeed, one of the primary motivations for this work is to provide a similar opportunity for Finnish NLP research. This motivation has affected both the choice of the scheme and the target size of the corpus, as will be discussed later.

2 Related work

As stated earlier, there is no publicly available treebank of general Finnish. The only treebank we are aware of is that of Haverinen et al. [4] who have applied the SD scheme to Finnish intensive care nursing narratives, producing a treebank of 1019 sentences. This treebank, however, is not publicly available due to patient privacy issues.

Also other NLP resources for Finnish are scarce. The only broad-coverage full syntactic parser for Finnish is the closed source commercial parser Connexor Syntax.[1] Other NLP tools, particularly targeted at morphological analysis, include FinTWOL and FinCG,[2] a morphological analyzer and a Constraint Grammar parser which resolves morphological ambiguity [5, 6], both commercial products. In addition, a rule-based parser has been developed by Laippala et al. [7], particularly targeting the language used in nursing narratives in a Finnish intensive care unit. This parser is, however, restricted to the very specific vocabulary and syntax typical for this domain.

Apart from the nursing narrative corpus of Haverinen et al., there is a second treebank with SD as its native annotation scheme, BioInfer [13]. It is an English-language corpus of 1100 sentences from research article abstracts focusing on protein-protein interactions. In addition to these two corpora, any treebank

[1]`http://www.connexor.eu`
[2]`http://www.lingsoft.fi`

annotated in the Penn Treebank [9] scheme can be automatically converted to the SD scheme using the method and tools[3] of de Marneffe and Manning [10].

3 Adaptation of the SD scheme to Finnish

In this section, we introduce our modifications to the Stanford Dependency scheme. Sections 3.2 and 3.3 discuss Finnish-specific adjustments, while Sections 3.4 through 3.7 consider more general modifications. Due to space limitations, the original SD scheme will only be discussed briefly, and the reader is referred to the work of de Marneffe and Manning [1] for a thorough description.

3.1 The Stanford Dependency scheme

In the SD scheme, the syntactic structure of a sentence is represented as a directed graph of labelled dependencies. The latest scheme version [1] defines 55 hierarchically arranged dependency types, capturing both syntactic and semantic relations. There are four different representation variants, in which different sets of dependencies are present. In the basic variant, used in the current annotation, the analyses are trees and generally include only syntactic dependencies. Other variants define a number of additional, semantically motivated dependency types that are present in addition to the basic syntactic dependencies. These variants thus result in non-tree structures that may even contain directed cycles.

The scheme is designed to be application-oriented and has indeed proved its usefulness in a number of NLP methods (for an extensive list, see the review by de Marneffe and Manning [10]). These successful applications have also contributed to our decision of using the scheme in this work, as has the encouraging observation that the SD scheme would seem to be suitable at least for clinical Finnish, as reported by Haverinen et al. [4].

Haverinen et al. adapted the SD scheme to clinical Finnish by introducing several new dependency types that address the most common Finnish syntactic structures that the SD scheme could not naturally represent: inflected nominal modifiers, adpositional phrases, and certain passive structures (for details, see [4]). These modifications apply with no further changes also to general Finnish, and, in the following, we discuss our additional adaptations of the scheme.

3.2 Genitive objects

In Finnish, a noun with a verb counterpart or a nominalization of a verb can have an object, called the *genitive object*. This resembles the English phenomenon where a gerundial noun takes an object in front of it, as in *ship building*, except that the genitive case is not used in the English structure. In English, nominal pre-modifiers

[3]`http://nlp.stanford.edu/software/lex-parser.shtml`

such as the above are considered syntactic compounds and are marked *nn* in the SD scheme.

On the surface, genitive objects are identical to possessive modifiers, both being nominal pre-modifiers in the genitive case. There is, however, a clear semantic difference between these two. For instance, the possessive interpretation of *laivan rakentaminen* (*ship+genitive building*) would mean that the ship itself is doing the building, whereas the genitive object interpretation would mean that the ship is being built. In order to maintain this semantic distinction, it is necessary to establish a new dependency type, *gobj*, for genitive objects.

3.3 Finnish copulas

The SD scheme reserves a special treatment for copula structures: the predicative of a copular clause is the head and the copular verb its dependent. In all other cases, the finite verb acts as the head. This is motivated from a multilingual point of view, as not all languages have an overt copular verb. Further, particularly in telegraphic style, the copular verb can often be omitted even in those languages that do. This treatment of copula structures, however, requires an exact definition of the class of copular verbs and predicatives.

The SD scheme uses a list of English copular verbs defined in the Penn Treebank, including, among others, *to be*, *to resemble* and *to become*. According to Finnish Grammar [3, §891], the only Finnish copular verb is *olla* (*to be*), and all clauses with *olla* as the main verb can be classified as copular. This includes clauses where the predicative is inflected in a local case, such as *Paketti on Oulusta* (*The_package is from_Oulu*). However, if a structure such as this one is accepted as copular, a sentence with several possible predicatives, such as *Paketti on Oulusta ystävältäni* (*The_package is from_Oulu from_my_friend*) can easily be formed. Such a structure has no obvious dependency representation in the SD scheme, since the clause would have two head words. Another problem related to the predicative cases is that of distinguishing the copular verb *olla* (*to be*) and other, non-copular verbs that take as their argument a noun inflected in the same case as the argument of the verb *olla*. Consider, for example, *olla laulajana* (*to_be singer+essive*), *toimia laulajana* (*to_act as_singer+essive*) and *työskennellä laulajana* (*to_work as_singer+essive*). All three examples have the same surface syntactic structure, yet for instance the third example is certainly not a case of copula.

To avoid the class of copulas becoming unnecessarily broad, and syntactically and semantically diverse, we only allow nominative and partitive cases for noun and adjective predicatives, which permits us to restrict copular structures to those that include the only Finnish copular verb, *olla*. In addition to nouns and adjectives, for instance adverbs and even full clauses can act as predicatives. Our solution, including our use of the separate copula subject type, *nsubj-cop*, is similar to that in the clinical treebank of Haverinen et al., although some of the most problematic cases do not occur in the clinical language. For an illustration of our analysis of Finnish copula structures, see Figure 1.

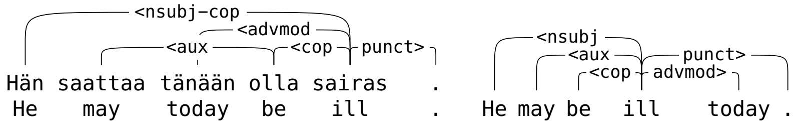
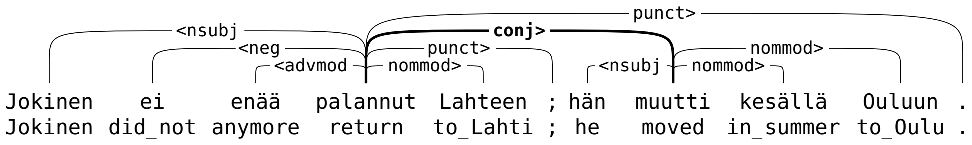


Figure 1: Finnish copula structures (left) as compared to those of English (right). Note that the copula acts as the head for the possible auxiliary which can sometimes cause non-projective structures. Also note the use of the *nsubj-cop* dependency type.



punct>
<nsubj
conj>
<neg
punct>
nommod>
<advmod
nommod>
<nsubj
nommod>
Jokinen ei enää palannut Lahteen ; hän muutti kesällä Ouluun .
Jokinen did_not anymore return to_Lahti ; he moved in_summer to_Oulu .

Figure 2: Implicit clausal coordination. The example sentence could be translated as "Jokinen did not return to Lahti anymore; he moved to Oulu in the summer."

3.4 Independent clause coordination

Independent clauses can be coordinated without a conjunction, as in *Lapset pyöräilivät kouluun; aikuiset ajoivat töihin* (*The children cycled to school; the adults drove to work*). The SD scheme analyzes such implicit coordination as parataxis and defines the corresponding dependency type. We, however find these structures functionally and semantically similar to explicit coordinations and thus also annotate them similarly. This is particularly natural in the SD scheme which analyzes conjunctions as mere dependents of the first coordinated element, making implicit and explicit coordinations differ only in the presence or absence of this single dependent. The *parataxis* type is then reserved for other types of parataxis such as reporting clauses. In this respect the scheme also diverts from that used by Haverinen et al., who defined a separate dependency type, *sdep*, for implicit clause coordination. Our analysis is illustrated in Figure 2.

3.5 Infinite clausal complements

The original SD scheme does not distinguish between finite and infinite clausal complements, but uses the type *ccomp* for both. For instance, in the structures *Sanoin, että pallo katosi* (*I_said that the_ball disappeared*) and *Estin palloa katoamasta* (*I_prevented the_ball from_disappearing*), the complements *että pallo katosi* (*that the_ball disappeared*) and *palloa katoamasta* (*the_ball from_disappearing*) would both be analyzed as *ccomp* in the original SD scheme. The *iccomp* dependency type enables the distinction of these two structures, which would otherwise not be possible without morphological information that, currently, is not present in the treebank.

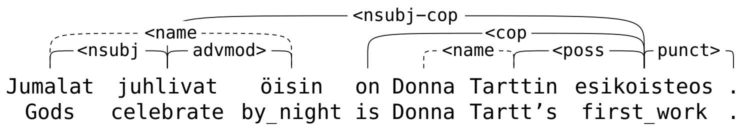

<nsubj-cop
<name
<cop
<nsubj
advmod>
<name
<poss
punct>
Jumalat juhlivat öisin on Donna Tarttin esikoisteos .
Gods celebrate by_night is Donna Tartt's first_work .

Figure 3: *Jumalat juhlivat öisin* (*Gods celebrate by night*) is a named entity with an inner syntactic structure and is thus given a full syntactic analysis, including the correct head word. *Donna Tarttin* is only marked as a multi-word unit with no further analysis. The technical dependency *name* is used to delimit named entity boundaries.

3.6 Named entities

Multi-word named entities, such as names of people, cities, books, and movies, are frequent in general language. These elements are problematic in a number of ways: often, but not always, they lack an obvious inner syntactic structure, despite consisting of several words, as for example *Carl Gustaf Emil Mannerheim*, or they may also be in another language, like *Västra Finnholmen*. An example of a name that does have a complex inner structure and is in Finnish would be the name of the book *Taistelu sosiaaliturvasta — ammattiyhdistysväen toiminta sosiaaliturvan puolesta 1957–1963* (in English *The battle for social security — trade union members' actions for social security 1957–1963*).

All multi-word names are annotated as single units whose rightmost word acts as the head in the dependency tree. In addition, Finnish names that do have an inner syntactic structure are given a full dependency annotation and their correct head word is identified (see Figure 3 for an illustration). This approach thus leaves open two options for treating Finnish named entities with inner structure. One possibility is to discard the annotation of the inner structure and consequently treat the named entities as single units. The other alternative is to preserve these entities as subtrees in the syntactic structure. The choice will likely be application-dependent.

3.7 Gapping and fragments

Gapping, a form of ellipsis where a governing element is omitted to avoid repetition while its dependents are not, poses an annotation problem. For instance, in *minä söin jäätelöä ja sinä salaattia* (*I ate ice cream and you salad*), the elided verb is necessary to construct a tree that correctly reflects the meaning of the sentence. A similar case is that of fragments, such as *Presidentti Kiinaan* (*The President to China*), where the head word of the clause is absent.

In order to be able to construct an analysis for such cases, we insert a *null* token into the sentences to represent the missing head word. In the case of gapping, where the antecedent of the elided element is present earlier in the sentence, we further include a semantic dependency, *ellipsis*, to relate the antecedent and the *null* token. In the case of fragments, no antecedent is present in the sentence and consequently

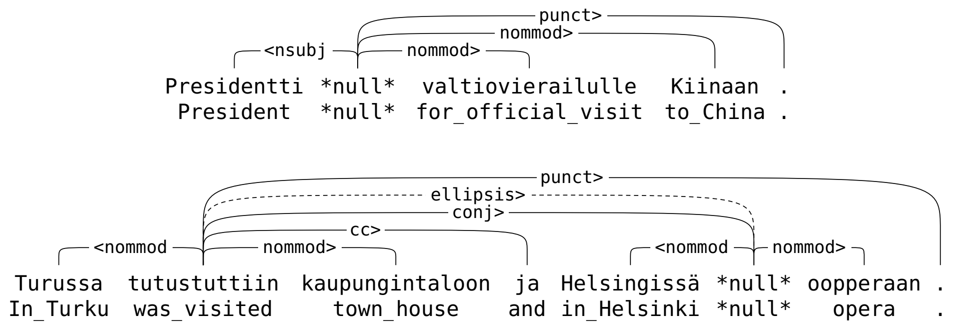

punct>
nommod>
<nsubj
nommod>
Presidentti *null* valtiovierailulle Kiinaan .
President *null* for_official_visit to_China .
punct>
ellipsis>
conj>
cc>
<nommod
nommod>
<nommod
nommod>
Turussa tutustuttiin kaupungintaloon ja Helsingissä *null* oopperaan .
In_Turku was_visited town_house and in_Helsinki *null* opera .

Figure 4: Null tokens in the case of fragments (top) and gapping (bottom). Note the semantic dependency *ellipsis*. The fragment sentence could be translated as "The President for official visit to China" and the ellipsis sentence as "The town house was visited in Turku and the opera in Helsinki."

the *ellipsis* dependency is not used. Figure 4 illustrates the usage of *null* tokens.

Note that the *null* tokens are only used to stand for missing governors. Consequently, other elements that do not generally act as governors in the SD scheme, such as missing copula verbs and auxiliaries, are not represented by *null* tokens, neither are other forms of ellipsis.

4 Construction of the Treebank

In this section, we describe the ongoing work on the Finnish dependency treebank itself.

4.1 Treebank text

In constructing the treebank, we use randomly selected articles from the Finnish Wikipedia. All articles that do not exceed 75 sentences are annotated in their entirety, excluding parts that do not have enough syntactic structure to annotate, such as bulleted lists of single words, section headings and figure captions. Longer articles are truncated at 75 sentences, to keep the treebank from becoming biased towards a single article topic.

Currently, we have completed the annotation of 60 articles, comprising 711 sentences and 10217 tokens, of which 8801 are non-punctuation. Thus, on average, one article is 11 sentences long and one sentence contains 14 tokens. The length of articles varies substantially — the longest article in the currently annotated part is 61 sentences long, while the shortest contains only one sentence. Out of all sentences in the treebank, 27 (3.8%) are non-projective. For comparison, Haverinen et al. report the proportion of non-projective sentences in the clinical treebank to be 2.9%. The currently existing annotation, subject to further changes, is available at `http://bionlp.utu.fi/fintreebank.html` to illustrate the annotation scheme.

4.2 The Annotation process

In our annotation work, we use a custom annotation tool, which will be made publicly available together with the treebank. It includes the basic abilities necessary for dependency annotation, along with search abilities and the possibility to mark a dependency for later discussion, or label a sentence as dubious or ungrammatical.

We have started the annotation process by first annotating 562 sentences (47 articles) in trial annotations. Each sentence was first annotated by one annotator and the annotation was then jointly inspected by the whole group. Authoritative decisions were made for all problematic cases found at this stage, and the already existing annotation was modified as necessary to ensure its consistency.

After the trial annotations, full double annotation has been started. That is, each sentence is first independently annotated by two annotators and all differences are then jointly resolved. The decisions made at this double annotation stage lean on the authoritative decisions made after the trial annotations. The current number of double annotated sentences is 149 (13 articles). Due to the currently small number of these sentences, we do not report an inter-annotator agreement at this stage, as this figure would not be representative. Inter-annotator agreement in the double annotation will be measured regularly throughout the annotation process to estimate the annotation quality and will be reported with the final release of the corpus.

5 Characteristics of Wikipedia text

The text in Wikipedia articles is sometimes thought to be of poor quality with respect to grammaticality. To determine some properties of the Wikipedia language, we have conducted a small-scale analysis of the currently annotated sample, estimating the proportion of spelling and grammar errors.

We assess the amount of spelling errors in the text by manually inspecting all words that FinTWOL,[4] a broad-coverage morphological analyzer, failed to recognize. Of the 1034 (10.1% of all tokens) unrecognized tokens, only 6 (0.6‰) were obvious misspellings, the remaining being most commonly names, foreign words, numerical expressions, untypical punctuation symbols, abbreviations, etc.

To estimate the level of ungrammaticality in the Wikipedia text, each sentence was assessed independently by three native speaker annotators, and marked *grammatical*, *questionable* or *ungrammatical*. All sentences not judged grammatical by at least two of the three annotators were further manually analyzed to determine the type of error they contained. The results of this manual analysis are given in Table 1. The vast majority of sentences, 691 out of 711 (97.2%), were judged grammatical by at least two annotators; 627 (88.2%) were judged grammatical unanimously. Further, 18 sentences (2.5%) were judged questionable and

[4]`http://www.lingsoft.fi`

Mistake type	Frequency
Fragment	6
Relative clause error	4
Compound error	3
Translation error, anglicism or colloquial	6
Inflection error	2
Coordination error	2
Total	23

Table 1: Results of the manual analysis of grammar errors. Note that the total number of errors is greater than the total number of ungrammatical and questionable sentences, as some sentences had more than one error in them.

2 (0.3%) ungrammatical. Out of the 20 sentences not judged grammatical, only one was downright incomprehensible. Fragments are among the most common cases judged questionable or ungrammatical, as are translation errors, anglicisms and colloquial language.

In general, many sentences judged as questionable were colloquial rather than strictly erroneous. Examples of such colloquial structures, which would in some contexts be judged ungrammatical, can be a sizeable asset for example when building a parser targeting text produced by non-professional writers. To conclude, we find the overall quality of the Wikipedia text, in terms of grammaticality and correct spelling, clearly acceptable.

6 Conclusions and future work

In this paper, we have presented first results of an ongoing effort to build a treebank of the Finnish language. First, we demonstrate that the Stanford Dependency scheme is applicable to general Finnish with only minor modifications. Many of these modifications have previously been introduced by Haverinen et al. [4] who applied the SD scheme to Finnish nursing narratives. Second, we assess the grammaticality of the Finnish Wikipedia language and find it, maybe somewhat surprisingly, clearly acceptable. In addition to the obvious benefit that Wikipedia text is freely available under an open license, it may also be an asset for a number of real-world applications that the language found in the articles can be colloquial and is not necessarily produced by professional writers. Currently, the treebank consists of 60 fully annotated articles, comprising of 711 sentences. The annotation is available at `http://bionlp.utu.fi/fintreebank.html`.

The primary goal of the project is to create a freely available treebank large enough for the induction of a broad-coverage statistical parser as well as the development of natural language processing methods in general. The first and most important future work direction is thus naturally to increase the size of the corpus.

Currently, we aim at annotating roughly 10,000 sentences, that is, about 140,000 tokens, a treebank size shown to be sufficient to induce an accurate statistical parser for a number of languages [11]. The performance and learning curve of the induced parser and other NLP methods that use the treebank will help to determine its final size.

A second, more long-term direction is to further enhance the annotation of the treebank by providing a layer of more detailed semantic analysis, for example using an SD scheme variant that also includes semantically oriented dependency types. In this layer, it would also be possible to deepen the annotation of elliptic structures by marking also omission of non-head elements. This will require further modifications to the SD scheme which does not prescribe any treatment of ellipsis. Thirdly, the possibility to provide morphological and POS information for the treebank using an existing analyzer for Finnish will be investigated.

Acknowledgements

We would like to thank Lingsoft Ltd. for making FinTWOL available to us. This work has been supported by the Academy of Finland and Turun Yliopistosäätiö.

References

[1] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies manual. Technical report, Stanford University, September 2008.

[2] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL 2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09*, pages 1–18, 2009.

[3] Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja-Riitta Heinonen, and Irja Alho. *Iso suomen kielioppi / Grammar of Finnish*. Suomalaisen kirjallisuuden seura, 2004.

[4] Katri Haverinen, Filip Ginter, Veronika Laippala, and Tapio Salakoski. Parsing clinical Finnish: Experiments with rule-based and statistical dependency parsers. In *Proceedings of NODALIDA'09, Odense, Denmark*, 2009.

[5] Fred Karlsson. Constraint Grammar as a framework for parsing unrestricted text. In *Proceedings of COLING'90*, pages 168–173, 1990.

[6] Kimmo Koskenniemi. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[7] Veronika Laippala, Filip Ginter, Sampo Pyysalo, and Tapio Salakoski. Towards automatic processing of clinical Finnish: A sublanguage analysis and a rule-based parser. *International Journal of Medical Informatics, Special Issue on Mining of Clinical and Biomedical Text and Data*, 2009. In press, available in online version only.

[8] Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114, 1998.

[9] Mitchell Marcus, Mary Ann Marcinkiwicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2), 1993.

[10] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies representation. In *Proceedings of COLING'08, Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, 2008.

[11] Joakim Nivre. Deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[12] Joakim Nivre. Sorting out dependency parsing. In *Proceedings of GoTAL'08*, pages 16–27, 2008.

[13] Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP'07*, pages 25–32, 2007.

Treebank Analysis and Search Using an Extracted Tree Grammar

Seth Kulick and Ann Bies

Linguistic Data Consortium
University of Pennsylvania
{skulick,bies}@ldc.upenn.edu[*]

November 4, 2009

Abstract

We describe here a new approach to the problem of analyzing and comparing two sets of trees that contain annotation for the same data. This is an important problem both for evaluating inter-annotator consistency during treebank construction and also for evaluating parser output as compared to the gold trees. Our approach is based on a decomposition of the trees into small syntactic chunks, inspired by work in Tree Adjoining Grammar. This allows queries to be stated in more meaningful syntactic units, and the resulting system produces confusion matrices showing disagreements on these units across the two sets of trees. There is also a significant potential speed advantage for treebank search, since duplicate information is removed from the treebank, and a search for the syntactic chunks simply becomes a search on integers associated with each sentence.

1 Introduction

A crucial issue when constructing a treebank is to ensure consistency in annotation decisions among the annotators. This is usually done by having multiple annotators annotate the same file, and then comparing the different annotations of that file. One way in which this has been done is to use the same metrics as used for scoring parser output against gold trees. This makes sense, since the two problems can

[*]We would like thank Aravind Joshi, Anthony Kroch, Mitch Marcus, and Mohamed Maamouri for many useful discussions. This work was supported in part by the Defense Advanced Research Projects Agency, GALE Program Grant No. HR0011-06-1-0003. The first author was also supported under the GALE program, DARPA/CMO Contract No. HR0011-06-C-0022. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

both be viewed as the same, as comparing two sets of trees to determine how well one matches against the other.

For example, the recent revision of the Penn Arabic Treebank [11] reported inter-annotator agreement (IAA) using the `evalb` program.[1] `evalb` determines a single score for the two sets of trees, by comparing matching constituent brackets across the two sets of trees.[2] However, while the `evalb` score can be viewed as a very rough approximation to the consistency of annotation, it is of little use in the important task of pinpointing where the annotators are disagreeing. To some extent this can be improved by breaking down the bracketing scores by bracket type (i.e., an `NP` score, an `SBAR` score, etc.). However, this is still just an approximation to the sorts of decisions actually made during annotation. One other approach that has been taken is to break down the two sets of trees into single-level head-dependency relations, as done for English in [5] and for Arabic in [10, 11].

There are of course systems available for searching a treebank corpus for various structures - e.g., [7, 12, 13]. These are extremely useful and heavily used (among other purposes) in corpus construction for identifying illegal structures that were mistakenly annotated. However, they also do not allow the comparison of structures across two sets of trees. Also, searches are stated in terms of individual nodes in the phrase-structure trees. While this allows great flexibility in searching for structures, it requires a certain amount of trickery to translate a search for meaningful syntactic units into the combination of single nodes.

Currently the analysis of disagreements among annotators in IAA annotation can only be done satisfactorily by painstaking and time-consuming manual comparison of the sets of trees, if the goal is to understand the underlying annotation decisions that led to the reported errors.

We present here an approach to a system for analyzing and evaluating two sets of trees in such a way as to allow search explicitly on these structures of interest, returning an analysis of differences on these structures. We break the trees down into meaningful syntactic units and search and evaluate based on these units, therefore returning information which is more aligned with the decisions made during annotation or mimicked during parsing. This approach is rooted in the long line of research on Tree Adjoining Grammar [9] which has been heavily used for various purposes in NLP but not previously for the purposes described here.

In Section 2 we discuss the particular data set we are using, which provides the examples of tree extraction and query search in the following sections. Section 3 describes the process of decomposing the full trees into the core syntactic units. This is similar to previous work in this tradition, although with some novel aspects, such as including the function tags in the extracted trees. In Section 4 we discuss how search on a treebank is reconceptualized and implemented to work on the tree decomposition of the full trees. In Section 5 we run through some examples of queries and results, and demonstrate the utility of this approach. Section 6 is the

[1] `http://nlp.cs.nyu.edu/evalb/`

[2] It also produces a few secondary numbers, such as the number of crossing brackets.

conclusion and discusses several possibilities for future work.

2 Data Set

There are several corpora under construction which are appropriate candidates for utilizing the work described here. We focus in this paper on the Penn Arabic Treebank, which has recently undergone a substantial revision [11] in its guidelines and for which more data is being annotated. Further, while there have been improvements in Arabic parsing ([10, 11]), there are still many questions as to which aspects of the tree structure are accurately recovered by the parser and which remain problematic.

Given the choice between working with a limited number of IAA files, and the more substantial number of trees available from parsing work, we have chosen to work first with the parse files. This is mostly because of the greater number of samples available for the development of this approach. As discussed above, it is basically the same problem regardless of which pairs of trees we work with.

The corpus we are using consists of the recent Arabic Treebank revisions of parts ATB1, ATB2, and ATB3.[3] For the parsing work, we used a previously-proposed train/dev/test split.[4]

The problem of segmentation of Arabic text into the tokens that match the gold tokens is not a simple one, and previously published work on parsing Arabic (e.g., [10, 11]), simply assumes gold tokenization for input to the parser. We instead use a morphological tagger and tokenizer [15], and for parsing use the Bikel parser.[5] [6]

3 Elementary Tree Extraction

As discussed above, we are aiming for an analysis of the trees that is directly expressed in terms of the syntactic constructions that annotators have in mind during annotation, or are mimicked during parsing. Towards this end we utilize ideas

[3]LDC2008E61 (Arabic Treebank Part 1 v4.0), LDC2008E62 (Arabic Treebank Part 2 v3.0), and LDC2008E22 (Arabic Treebank Part 3 v3.1), respectively.

[4]`http://nlp.stanford.edu/software/parser-arabic-data-splits.shtml`

[5]`http://www.cis.upenn.edu/~dbikel/software.html`

[6]There is an important issue here for parser evaluation that we can only discuss briefly. The `evalb` program depends on the two sets of trees having exactly matching tokenizations, so that the corresponding constituent spans can be compared. The spans cannot be compared when the tokenizations do not match. In fact, while the tagger has an overall tokenization accuracy of about 98.5%, even this level of accuracy is catastrophic for `evalb`, since out of 1739 sentences in the dev section (of length $<=$ 40), 338 differ in tokenization and so cannot be evaluated, and for this reason we do not present a parsing score. See [17] for a discussion of a similar problem for Chinese parsing evaluation. A good candidate for overcoming this problem is the Sparseval software [14], which allows the separate specification of how tokens are aligned. This is somewhat orthogonal to the main concerns of this paper, but we note here that this issue does not prevent the evaluation and analysis under our system. See footnote 13 for more detail on this issue.

```
===============================
TOKENS
===============================
<0> wa  {and}                            <1> jar+at {occur+it/they/she]}
<2> Al+masiyr+ap {the+march+[fem.sg.]}   <3> Al+>uxoraY {the+another}
<4> fiy {in}                             <5> muxay~am {refugee camp}
<6> jabAlyiA {Jabaliya}                  <7> li {for/to}
<8> lAji}+iyna {refugee+[masc.pl.gen.]}  <9> $amAl {north}
<10> gaz~+ap {Gaza+[fem.sg.]}            <11> bi {by/with}
<12> mu$Arak+ap {participation+[fem.sg.]}  <13> Ea$ar+At {scores+[fem.pl.]}
<14> Al+>aTofAl {the+children}

================================================================
FULL TREE
================================================================

(S (CONJ <0>wa)
   (VP (PV+PVSUFF_SUBJ:3FS <1>jar+at)
       (NP-SBJ (DET+NOUN+NSUFF_FEM_SG <2>Al+masiyr+ap)
               (DET+ADJ <3>Al+>uxoraY))
       (PP-LOC (PREP <4>fiy)
               (NP
                   (NP (NOUN <5>muxay~am})
                       (NP (NOUN_PROP <6>jabAliyA)))
                   (PP (PREP <7>li)
                       (NP (NOUN+NSUFF_MASC_PL_GEN <8>lAji}+iyna)))
                   (NP-LOC (NOUN <9>$amAl)
                          (NP (NOUN_PROP+NSUFF_FEM_SG <10>gaz~+ap)))))
       (PP-MNR (PREP <11>bi)
               (NP (NOUN+NSUFF_FEM_SG <12>mu$Arak+ap)
                   (NP (NOUN_NUM+NSUFF_FEM_PL <13>Ea$ar+At)
                       (NP (DET+NOUN <14>Al+>aTofAl)))))))

================================================================
EXTRACTED ETREE INSTANCES
================================================================
#     ETREE TEMPLATE                    ANCHORS
1     A1                                (A1) <0> CONJ wa
2     (S (VP A1 NP[t]-SBJ^))            (A1) <1> PV+PVSUFF_SUBJ:3FS jar+at
3     (NP A1)                           (A1) <2> DET+NOUN+NSUFF_FEM_SG Al+masiyr+ap
4     A1                                (A1) <3> DET+ADJ Al+>uxoraY
5     (PP[b]-LOC A1 NP^)                (A1) <4> PREP fiy
6     (NP A1 (NP A2))                   (A1) <5> NOUN muxay~am
                                        (A2) <6> NOUN_PROP jabAliyA
7     (PP A1 NP^)                       (A1) <7> PREP li
8     (NP A1)                           (A1) <8> NOUN+NSUFF_MASC_PL_GEN lAji}+iyna
9     (NP[b]-LOC A1 (NP A2))            (A1) <9> NOUN $amAl
                                        (A2) <10> NOUN_PROP+NSUFF_FEM_SG gaz~+ap
10    (PP[b]-MNR A1 NP^)                (A1) <11> PREP bi
11    (NP A1 (NP A2 (NP A3)))           (A1) <12> NOUN+NSUFF_FEM_SG mu$Arak+ap
                                        (A2) <13> NOUN_NUM+NSUFF_FEM_PL Ea$ar+At
                                        (A3) <14> DET+NOUN Al+>aTofAl
```

Figure 1: An example of Tree Decomposition resulting in extracted elementary trees (etrees) (Translation of Arabic sentence: Another march occurred in the north Gaza Jabaliya refugee camp for refugees with the participation of scores of children.)

from a line of research on decomposing full trees in a treebank into smaller syntactic chunks. Usually based around Tree Adjoining Grammar (TAG) or some variant (loosely referred to as "tree grammars"), this work aims to identify the smaller trees that are the "building blocks" of the full trees of that treebank, and that are then used for such purposes as training parsers or as a basis for machine translation systems [3, 4, 16]. However, this approach has not been utilized for searching within a treebank, as far as we know.

As in the earlier TAG work we use head rules to decompose the full trees and then extract out the "elementary trees", which are the small syntactic chunks. For our grammar we use a TAG variant with tree-substitution, sister-adjunction, and Chomsky-adjunction ([4]). We do not have space here to review these basic aspects of the extraction in detail, but instead we give an example in Figure 1, and highlight some aspects of our tree extraction that we feel are worthy of note.[7]

The full tree is shown in the middle of Figure 1.[8] Each token is listed as `(POS <index> word)`, where `<index>` is the index of the word in the sentence. To avoid cluttering up the tree structure, we include at the top a separate listing of the glosses for each word.

The extracted elementary trees are shown at the bottom. Each decomposed tree has a particular tree structure, and it is possible (indeed, it is the entire reason for this approach) that the same tree structure is used in more than one decomposed tree fragment. We call each such elementary tree structure an "etree template", and a particular instance of that template, together with the "anchors" (tokens) used in that instance of that template, is called an "etree instance" ("etree" is short for "elementary tree").

For example, the template `(S (VP A1 NP[t]-SBJ^))` is used once in this tree decomposition (although many times in the entire corpus), where `A1` is the anchor of the template, which is associated with a particular word in an etree instance that uses this template. In this case, instance #2 uses this template, and the anchor is the verb at index `<1>`. The `^` indicates that the `NP[t]-SBJ^` node is a substitution node, meaning that another etree instance substitutes into it to reform the original full tree (in this case, etree instance #3).

The template `(NP A1)` is used in two etree instances, #3 with anchor `<2>` and #8 with anchor `<8>`. Templates can have more than one anchor. For example, the template `(NP[b]-LOC A1 (NP A2))` is an example of a two-level *idafa* structure, an extremely common structure in Arabic, in which two or more tokens form a tight syntactic unit. (In this particular case, the entire structure projects with a `LOC` function tag as well.) Etree instance #9 uses this template, with two anchors, the words at indices `<9>` and `<10>`. Instance #11 is an example of a three-level idafa.

A fundamental idea of this approach (as in all TAG-related work) is that the modifiers are separated from non-recursive structures. For example, the two-level

[7]See [8] for an earlier and somewhat different approach to extracting a tree grammar from the Arabic Treebank.

[8]Throughout this paper we use the Buckwalter Arabic transliteration scheme [2].

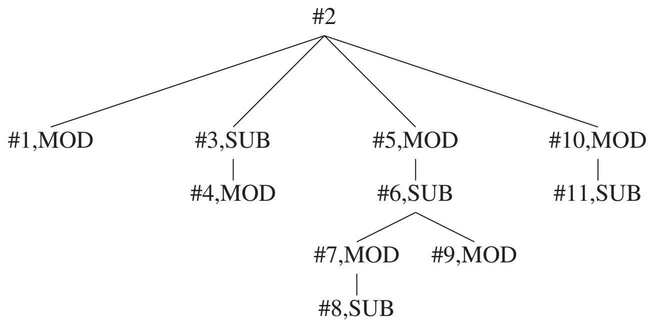


Figure 2: The derivation tree for the extraction in Figure 1

idafa at indices `<5>,<6>` is separated from its modifier `PP` and `NP-LOC` sisters.[9] Some modifiers result in etree templates that are just a single anchor, with no structure. For example, the template `A1` is used for instance #1, of just the typical sentence-initial conjunction `wa`, and also for the adjective `<3>`, which is separated out from the noun it modifies (`<2>`).

It is important to note that each etree instance has at least one anchor, and every tree token is an anchor for some etree instance. It therefore becomes possible to examine the properties of an etree instance that a tree token is the anchor for, and to compare corresponding instances for the same tree token across two different annotations, such as gold/parser-output trees or between the trees for two different annotators. This property (traditionally called "lexicalization" in the Tree Adjoining Grammar literature) is taken advantage of for query searching and comparison in Sections 4 and 5.

As usual in tree grammar extraction, the extraction process for each full tree produces not just the collection of etree instances, but also a "derivation tree" that is in effect a record of how the etree instances combine together again to form the original full tree. The derivation tree for the extraction just described is shown in Figure 2, in which the nodes of the tree correspond to the etree instances in Figure 1, and the `SUB` and `MOD` indicate the type of attachment (substitution and modification).[10] For example, the derivation tree shows etree instance #4 modifying instance #3, which in turn substitutes into instance #2.

Unlike all earlier work in tree decomposition that we know of, we also include function tags in the extracted trees. While a prominent feature of the Penn Treebank and the Penn Arabic Treebank, they have been mostly ignored in parsing (with some exceptions - e.g., [1, 6]) and in previous tree extraction work.[11]

[9]The "extra" NP that scopes around the span `<5,10>` in the full original tree is therefore missing from the extracted instances. If recreating the original tree from the extracted trees, it can be added back in to exactly recreate the original tree.

[10]This derivation tree is a bit of a simplification. The complete tree also includes the addresses in each etree template of the locus of substitution or modification, and for sister adjunction, the direction of modification. We leave out these details here.

[11]Except for the use of the function tags as a way to determine the argument/adjunct classification

Since the function tags are an important part of the annotation (see the arguments in [1, 6]), and since they are part of the annotation process (and so something to be checked in the inter-annotator agreement files), we include them in the tree decomposition. We break the tags up into two groups "syntactic" and "semantic". The TAG decomposition views the former as being imposed on a node from the top, and the latter as arising from the bottom, and this controls where the function tag for a node from the original tree is placed in the extracted trees. For example, the `SBJ` function tag (a syntactic tag) in the full tree is placed in the template for instance #2 (with the `[t]-SBJ` meaning that it is a "top" function tag), and it is *not* present on the template for instance #3, and so the word at index `<2>` in etree instance #3 only receives that label from substituting into the `NP[t]-SBJ^` node. In contrast, the `-LOC` and `-MNR` tags are semantic tags, and so "bottom" tags and appear in the extracted trees anchored by the head of the constituent the tag appears with in the original tree.

4 Treebank Search Using Elementary Trees

We carry out the extraction procedure described in the previous section to both versions, gold and parsed, of the dev section, using the data discussed in Section 2.[12]

For the two versions of the dev section, taken together, there are 141499 tokens, with 83196 etree instances, which need only 1551 etree templates. After this extraction process is completed, all of the tokens, etree templates, etree instances, and derivation trees are stored in a MySQL database for later search. We do not have space here to show the database schema, but it is organized with appropriate indexing to allow for very quick access to the etree instances for each sentence, and to the etree template each one in turn links to.

The search for particular syntactic structures, as represented by the etree templates, therefore becomes a search for indexed integers (with integers representing etree instances and etree templates). Searching for these elementary trees is therefore extremely fast, something we are taking advantage of for separate work on searching an entire corpus for syntactic structures, aside from the current issue of searching on two sets of trees together. From the perspective of database organization, the tree extraction can be perhaps be viewed as a type of database "normalization", in which duplicate information is placed in a separate table.

Before any queries are processed, the tree decomposition and database creation, as discussed above, is done. This is only done once, for the two full sets of

in various head-based parsing approaches, or in the earlier TAG-based tree extraction work. However, most parsing work has not been concerned with function tag recovery during parsing, and no TAG-based extraction work has included the function tags in the extracted trees.

[12]The extraction of the parse output requires that function tags be included in the output, as in either [1, 6]. This is because the function tags are used for the tree decomposition process. However, it does not currently use empty categories in the parse output, as in [6] and other work, although that will be incorporated in future work.

trees, and is used for all future query searches.

Currently queries are specified as conditions over elementary trees. We do not currently utilize the derivation tree in the search as well, to query on how etrees connect with each other. However, this is very important for future work, as discussed in Section 6.

Further, the queries can be logically grouped into sets, for cases in which we want to see how particular etree instances might satisfy one query instead of another. For example, we might want to see if the parser, or the annotators, are confusing `LOC` and `ADV` function tags, or treating a particular token as part of a three-level idafa instead of a two-level idafa.

We take advantage of the "lexicalized" property of the tree grammar, as discussed in Section 3, to allow us to construct confusion matrices showing how corresponding tokens across two different annotations compare with regard to satisfaction of the queries of interest. This is possible because we can associate each token with satisfaction results for various queries based on the etree instance that the tree token belongs to. Even queries examined on their own, without reference to other queries, form a 2x2 confusion matrix, in which the query is compared against the lack of satisfaction of that query. We show examples of these query results and confusion matrices in the next section.

For a given set of queries, the following sequence occurs:

- The etree templates are searched to determine which match a given query. It is possible (and likely) that a query will select more than one of the 1551 etree templates. For example, a query specifying the structure (SBAR WHNP S) will find cases of such SBAR trees regardless of what is below the S node (intransitive or transitive verb, verbal noun, etc.).

- The etree instances are then searched to determine which take a template that satisfies a given query.

- These first two steps result in the following situation. Each etree instance is categorized as satisfying a given query or not. Due to the "lexicalized" property of the tree grammar, as just discussed, each treebank token is linked to an etree instance, and so it immediately follows that for any treebank token, it is easy to determine whether it is in a syntactic context that satisfies a given query.

- The two sets of trees are gone through in parallel, token by token.[13] Each corresponding token in the two sets of trees is checked for which queries it satisfies. The query results are stored in confusion matrices as mentioned above.

[13]As noted in footnote 6, there is an issue here with differing tokenizations in the gold and parser data. More precisely, we are gathering data on the original whitespace-delimited tokens, which themselves are broken up into multiple tokens for the trees, perhaps differently for the gold and parsed data. This allows the matching of tokens across the two sets of trees, thus overcoming the problems with `evalb` mentioned in footnote 6. We do not show in this paper the two levels of tokens.

gold\ parsed	N	query 4	total
N		194	194
query 4	88	677	765
	88	871	959

Table 1: Confusion matrix showing results of query 4. `N` indicates absence of satisfaction of query 4. The cell `(N,N)` is not included since we are only interested in cases in which at least one query was satisfied.

gold\ parsed	N	1	2	3	total
N		6	0	131	137
query 1	16	28	0	8	52
query 2	3	0	1	1	5
query 3	215	2	0	313	530
total	234	36	1	453	724

Table 2: Confusion matrix showing results of queries 1, 2, and 3.

5 Data Set and Example Queries

We give four examples of searches that we are currently using. They are specified as partial tree information, and are matched against the etree templates. The `A$` indicates the anchor of an elementary tree that is associated with that query.[14]

1 - `(NP[b]-LOC A$)` - the anchor projects to a NP-LOC

2 - `(NP[b]-DIR A$)` - the anchor projects to a NP-DIR

3 - `(NP[b]-ADV A$)` - the anchor projects to a NP-ADV

4 - `(NP A (NP A (NP A$)))` - a three-level idafa

In addition, we consider query 4 by itself, resulting in a 2x2 confusion matrix for cases in which a three-level idafa was found vs. the absence of a three-level idafa. Queries 1, 2, and 3 are grouped together in a confusion matrix, along with absence of satisfaction of query 1, 2, or 3.

Table 1 shows the confusion matrix for query 4, the three-level idafa. The `N` row indicates the absence of query 4 in the gold tree, while the `N` column indicates the absence of query 4 in the parsed tree. We do not include the cell `(N,N)` because those would be cases in which the query under consideration is missing from both the gold and parsed trees for some token, and so are not relevant for this table.

The cell `(4,4)` indicates that there are 677 cases where both sets of trees agreed on the same three-level idafa. However, there are 88 cases where the gold tree had a three level idafa and the parsed tree did not, and 194 cases of the inverse. Figure 3 shows an example of an entry from cell `(N,4)` in Table 1, in which the token `<10>` in the parsed tree is the anchor of a three-level idafa, thus satisfying query 4, while the corresponding anchor in the gold tree does not. Note that it is a simple

[14]i.e., if an elementary tree has more than one anchor, we want only one anchor (word) to trigger that query, so that the elementary tree is not counted twice.

```
        GOLD TREE                                        PARSED TREE
        =========                                        ===========
(PP (PREP                                                  (PP (PREP
     <7>li,for/to)                                              <7>li)
    (NP (NOUN+NSUFF_MASC_PL_GEN                                (NP (NOUN
         <8>lAji}+iyna,refugee+[masc.pl.gen.])                      <8>lAji}+iyna)
(NP-LOC (NOUN+CASE_DEF_ACC                                         (NP (NOUN
         <9>$amAl+a,north/North+[def.acc.])                              <9>$amAl)
        (NP (NOUN_PROP+NSUFF_FEM_SG+CASE_DEF_GEN                        (NP (NOUN_PROP
             <10>gaz~+ap+i,Gaza+[fem.sg.]+[def.gen.])                           <10>gaz~+ap)
```

Figure 3: The token `<10>gaz~+ap` in the parsed tree is an example of a token that results in the cell `(N,4)` in Table 1, since it is the anchor of a three-level idafa while the corresponding token in the gold tree is not. At the same time, the token `<9>` in the gold tree results in an entry in cell `(1,N)` in Table 2 since it is the anchor of a `NP-LOC` structure, while the corresponding token `<9>` in the parsed tree is not.

matter in this approach to report on multi-level etree structures that are larger than just one-level relations in the trees.

Table 2 shows the confusion matrix for queries 1, 2, and 3. For example, the cell `(1,1)` indicates that there are 28 cases in which both the gold and parsed trees had an `(NP-LOC A)` structure for a token `A`. Cell `(1,3)` indicates that there are 8 cases in which the gold tree satisfied query 1 `(NP-LOC)` for some token, while the parser output tree satisfied query 3 `(NP-ADV)` for that same token.

Of particular interest to us is that cell `(1,N)` indicates that there are 16 cases in which the gold tree had a token projecting to `NP-LOC` while the parsed tree did not project to LOC, DIR, or ADV. One such case is the same pair of tree fragments in Figure 3, in which token `<9>` in the gold tree heads an `NP-LOC` while this is not the case in the parsed tree.[15]

These examples are of necessity just a sampling of the searches currently being used. The ability to define conditions on etrees and relate the results to particular tokens makes it easy to look for properties across the pairs of trees. It is extremely useful to be able to define queries on meaningful syntactic units, such as the idafas or, in other queries not shown here, properties of relative clause constructions, properties of etrees headed by verbs (such as valency), and so on. These types of searches are harder to do in the earlier work on tree analysis discussed in Section 1, which are limited to reporting on one-level head/dependency relations.

[15]This case is also of interest also because it shows that the parser is not taking advantage of morphological evidence showing that the three-level idafa analysis is wrong. The tight coupling of words in an idafa triggers various morpological and phonological effects, one of which is that the "n" in non-final words in the idafa is dropped, and so the suffix "iyna" in token `<8>` clearly indicates that the idafa in the parse output must be incorrect.

6 Conclusions and Future Work

We have described a new approach to treebank search that allows queries to be directly stated, and reports generated, using meaningful units of tree fragments. Future work will take place in three overlapping directions:

(1) The work has so far focused on Arabic parsed data. This needs to be extended for IAA analysis (which, again, is basically the same problem), and furthermore, for other languages, in particular English, both for parsing analysis and for current treebank construction.

(2) As discussed in Section 4, there are some notable potential speed advantages to this approach, and we intend to utilize and test this for the more standard problem of searching on a complete corpus for various configurations, in addition to the search on parallel trees as described here.

(3) However, by far the most important aspect of future work is extending the query definition to allow for searches on how etrees combine in the derivation tree. This will allow us to quantify various aspects of modifier attachment, always a concern for both annotation and parsing. Because we are using trees of meaningful syntactic structures as the basic units of search, we will be able to quantify the results concerning such questions as “how often do the annotators agree on the core structures, but disagree on the attachment of various modifiers into those structures?”

While searching in the derivation tree is necessary for fully expressing the searches we are interested in, it will add a layer of complexity to the search procedure. The worst-case scenario is that requiring arbitrary searches over the derivation tree will bring up here the usual issues concerning how to search trees in a database that we have so far avoided in this approach by “normalizing” the corpus by extracting out the etrees. However, we suspect that many of the required searches will need to examine only very local slices of the derivation tree, namely just parent/head/sister relations, which are those used to represent modification in the derivation tree. This should be the case because in the derivation tree, each node already represents a chunk of syntactic structure, instead of just a a single node. If this hypothesis holds, and arbitrary hierarchical searching on trees is avoided for queries, we will be able to search for queries we are interested in, while avoiding any significant speed penalty.

References

[1] Don Blaheta. *Function Tagging*. PhD thesis, Brown, 2003.

[2] Tim Buckwalter. Arabic morphological analyzer version 2.0. LDC2004L02, 2004. Linguistic Data Consortium.

[3] John Chen. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. PhD thesis, University of Delaware, 2001.

[4] David Chiang. Statistical parsing with an automatically extracted tree adjoining gramar. In *Data Oriented Parsing*. CSLI, 2003.

[5] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589–637, 2003.

[6] Ryan Gabbard, Seth Kulick, and Mitchell Marcus. Fully parsing the Penn Treebank. In *HLT-NAACL*, pages 184–191, 2006.

[7] Sumukh Ghodke and Steven Bird. Querying linguistic annotations. In *Proceedings of the Thirteenth Australasian Document Computing Symposium*, pages 69–72, Hobart, Australia, 2008.

[8] Nizar Habash and Owen Rambow. Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank. In *Traitement Automatique du Langage Naturel (TALN-04)*, Fez, Morocco, 2004.

[9] A.K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 69–124. Springer, New York, 1997.

[10] Seth Kulick, Ryan Gabbard, and Mitchell Marcus. Parsing the Arabic Treebank: Analysis and improvements. In *Proceedings of TLT 2006*. Treebanks and Linguistic Theories, 2006.

[11] Mohamed Maamouri, Ann Bies, and Seth Kulick. Upgrading and enhancing the Penn Arabic Treebank: A GALE challenge. In Joseph Olive, editor, *in progress for publication (book describing work in GALE program)*. 2009.

[12] Jiri Mirovsky. Netgraph - making searching in treebanks easy. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 945–950, Hyderabad, India, 2008.

[13] Beth Randall. Corpus search. http://sourceforge.net/projects/corpussearch/.

[14] B. Roark et al. Sparseval: Evaluation metrics for parsing speech. In *Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy, 2006.

[15] Rushin Shah. The LDC Standard Arabic Morphological Tagger. talk at Linguistic Data Consortium.

[16] Fei Xia. *Automatic Grammar Generation From Two Different Perspectives*. PhD thesis, University of Pennsylvania, 2001.

[17] Nianwen Xue. Evaluating the impact of Chinese word segmentation on syntactic parsing, 2009. Book chapter for Global Automatic Language Exploitation.

A Declarative Formalism for Constituent-to-Dependency Conversion

Torsten Marek Gerold Schneider Martin Volk

Institute of Computational Linguistics
University of Zürich

marek@ifi.uzh.ch gschneid@ifi.uzh.ch volk@cl.uzh.ch

Abstract

In this paper, we present a declarative formalism for writing rule sets to convert constituent trees into dependency graphs. The formalism is designed to be independent of the annotation scheme and provides a highly task-related syntax, abstracting away from the underlying graph data structures.

We have implemented the formalism in our search tool and used a preliminary version to create a rule set that converts more than 97% of the TIGER corpus.

1 Introduction

Syntactic structures are expressed either as constituent or as dependent structures. The fundamental differences between the two formalisms is that dependency is (1) endocentric, which means that the category of the parent node follows from the category of the child node, and (2) strictly binary, which means that the conversion of ternary rules (and n-ary if n > 2) is ambiguous. Debates about which formalism is theoretically more appropriate for linguistics are no longer topical, [4] has e.g. shown that X-bar grammar (Principles and Parameters) is equivalent to Dependency, but the practical problem of conversion persists. On the one hand, the theoretical discussions have led to successful mixed formalisms (e.g. HPSG, LFG). On the other hand, tackling the intricate details of a given annotation scheme is now more seen as a practical problem, where the devil is in the details. We present a declarative formalism and a tool that supports the linguist in writing conversion rules. The tool graphically displays conversions sentence-by-sentence and immediately updates rule changes. It also delivers errors and coverage reports, allowing a linguist to write a broad-coverage conversion in little time.

The paper is structured as follows: We summarize related work in section 2. The conversion formalism is introduced and illustrated with examples in section 3. We describe the performance of a sample conversion rule set that we have written for the German TIGER Treebank in section 4. We discuss future work in section 5.

2 Related Work

On an abstract level, conversion of a constituent tree into a dependency graph is an application of *graph transformation* [8], which is used in fields like compiler theory and software architecture validation. A graph transformation is a series of rewrite rules which are applied to a source graph to create a target graph. Our formalism is influenced by this idea, but while graph transformations are domain-oblivious, our formalism is coupled strongly to the domain of syntactic annotation as graphs. On a practical level, a number of conversion algorithms already exist.

2.1 MALTparse

[12] for example maps Penn Treebank constituents to dependency representations. As the majority of Penn Treebank constituents do not have functional labels, relation labels need to be assigned. In order of descending priority, the rules are as follows. (*mother = M, head = H, dependent = D, r = dependency relation label*)

1. if *D* is a punctuation category, *r* = P.
2. if *D* contains the function tag SBJ, *r* = SBJ.
3. if *D* contains the function tag PRD, *r* = PRD.
4. if *M* = *VP*, *H* is a part-of-speech tag and *D* = NP (without any function tag), *r* = OBJ.
5. if *M* = *VP*, *H* is a part-of-speech tag and *D* = VP, *r* = VC.
6. if *M* = *SBAR* and *D* = S, *r* = SBAR.
7. if *M* = *VP, S, SQ, SINV or SBAR*, *r* = VMOD.
8. if *M* = *NP, NAC, NX or WHNP*, *r* = NMOD.
9. if *M* = *ADJP, ADVP, QP, WHADJP or WHADVP*, *r* = AMOD.
10. if *M* = *PP or WHPP*, *r* = PMOD.
11. Otherwise, *r* = DEP.

This mapping is simple and reliable but leads to a less elaborate dependency formalism than e.g. the Stanford scheme [5]

2.2 Johannsson

[9] describes constituent-to-dependent conversion as a two stage process: head-selection (as most constituent representations are not endocentric) and function assignment (as most dependency relations are labeled). Head selection typically involves a set of Magerman rules [11]. The function assignment function is an extension of [12], addressing e.g. the distinction between object and adjunct, and raising long-distance dependency nodes.

2.3 Pro3Gres

[15] describes mapping as a complex task, which comes with some loss. In order to train the dependency parser Pro3Gres on Penn Treebank data, an involved mapping that has high precision but slightly incomplete recall, and that does not map all structural configurations is used. This leads to a mapping that delivers reliable relations but not fully connected trees, which is sufficient for the task of delivering statistical data for parser training. The annotation scheme is very similar to the Stanford scheme [7].

The structural configurations are queried with a large collection of tgrep queries, the majority of which are non-local, allowing access to lexical material and dealing with long-distance dependencies.

2.4 TiGerDB

There are several approaches converting the Penn Treebank, but there is less research on other Treebanks. Forst et. al [6] converted the German TIGER Treebank into a dependency representation, TiGerDB. The conversion had to be done semi-automatically, since TiGerDB has a richer annotation than the original data. Boyd et al. [1] address several issues in TiGerDB, which keep it from becoming a proper gold standard for dependency parsing and map it to a more surface-oriented analysis which does not include abstract nodes and is aligned with the original corpus tokens.

3 The Conversion Formalism

Developing a formalism for converting phrase structure into dependency trees that is independent of the annotation formalism, must strike a balance between expressivity on the one and task-relatedness on the other hand. If the formalism is too expressive, it might just as well be implemented as a framework or library for a general purpose programming language.

On the other hand, the formalism should have sufficiently expressive power, even for structures that can be arbitrarily large, like coordination constructions; and a syntax that captures the problems in constituent-to-dependency conversion, head identification and introduction of dependency edges.

3.1 Data Structures

In the conversion process, syntax trees, irrespective of the actual syntax formalism, are represented as directed graphs with typed nodes and edges. A graph G has a set of vertices (or nodes) V_G and a set of directed edges E_G. An edge going from p to c with $\{p,c\} \subset V_G$ is represented by $(p,c) \in E_G$.

Nodes and edges are typed attribute-value matrices (AVMs), with the actual types depending on the syntax and annotation formalism. There are several different types of edges, thus graphs need not be proper trees (i.e. one and only one node from which all others can be reached, no cycles), however the edges that define the *main structural layer* must satisfy the requirements for a tree.

3.1.1 Constituent Graphs

Based on the type system in [10], a constituent graph C consists of terminal and nonterminal nodes, representing words and linguistic phrases. The base type of all node types is the *feature record*. The main structural layer is defined by dominance edges, which define the phrase structure. All nodes but the root node have exactly one incoming dominance edge. If we refer to the origin and target node of a dominance edge, we use the terms *parent* and *child*.

The linear ordering of terminals in the sentence is annotated explicitly, which allows crossing dominance edges. This can be encoded using precedence edges between successive terminals, but for performance reasons each terminal has its positional index as a feature value.

3.1.2 Dependency Graphs

In contrast to constituent graphs, which are created from two different kinds of node types, a dependency graph D as defined in this paper contains only nodes representing words, again with explicit linear ordering.

The main structural layer is defined by dependency edges. The origin of a such an edge is called *head*, its target *dependent*. A word can have any number of outgoing dependency edges, but has at most one incoming dependency edge. Words without incoming edges are the root node and, depending on the formalism, punctuation and other non-word tokens.

Some formalisms for dependency graphs allow empty nodes, i.e. nodes that act as heads but do not correspond to any word in the sentence, thus resembling phrase nodes in constituent trees. In our conversions, we do not use empty nodes so far.

3.2 Conversion Process Constraints

Following the approach described in [9], conversion is carried out by identifying a head child for each phrase in the constituent tree and connecting the remaining children to the head using appropriate dependency edges. The conversion of a whole tree is carried out by repeated applications of *bound rules*, which define the rules for different kinds of linguistic phrases. The conversion process is *local* in the sense that only a phrase and its immediate children are considered during rule application. Nonterminal children are assumed to be opaque regarding their internal structure.

The conversion of a constituent graph C is *sound* if the generated dependency tree D satisfies the structural requirements described in section 3.1. The conversion is *complete* if each nonterminal node p is matched by a bound rule and if this rule handles each dominance edge emanating from p explicitly in the conversion process. The completeness criterion does not require that each dominance edge in E_c is converted into a dependency edge in E_D, cf. 3.3.2 for a justification of this relaxation.

Both constituent and dependency trees build structures on the exact same sentence material, thus the first step in the conversion is always to copy all terminal nodes from the constituent tree into the dependency tree.

3.3 Bound Rules for Nonterminals

Bound rules are the core of each rule set. They describe the conversion of a nonterminal node p and all its immediate children, i.e. all nodes $c \in V_C$ for which $(p,c) \in E_C$ holds. Example 1 shows the structure of such a rule.

(1) `context { <node> } { <body> }`

The keyword `context` is followed by the rule context `<node>`, which specifies the nonterminal nodes the rule is applied to. Here, we use the well-established syntax introduced by the TIGER query language [10], which can be used to query arbitrary acyclic directed graphs with feature structures as nodes. The context is currently limited to *node descriptions* and can thus only be used to select nodes based on local features. Extended contexts with structural constraints are discussed in section 5.1. The following list shows examples of node descriptions.

- `[cat="NP"]`
  Matches all nodes where the feature *cat* is `"NP"`
- `[cat=("CS"|"CNP")]`
  Matches all nodes where the feature *cat* is either `"CS"` or `"CNP"`

The second pair of curly braces contains the *body* (`<body>`) of a bound rule. The body is a series of actions, which are used to create the structure of the dependency graph, cf. section 3.3.2. They are applied to node variables which are introduced through quantifiers. The general form of a quantified expression is shown in example 2.

(2) `<quant> <var> { <child> } => <actions>;`

This expression states that the list of children of the current nonterminal node should be searched for nodes c that satisfy the local context given in `<child>`. The context is defined by a partial node description, which can be made more specific by constraining the label of the edge (p,c). Examples of child contexts are:

- `NK [pos="NN"]`
  Any word with part-of-speech tag `NN` and incoming dominance edge label `NK`.
- `AC [FREC]`
  Any node with incoming dominance edge label `AC`.

3.3.1 Quantifiers

The quantifier `<quant>` determines the way the individual nodes that satisfy the context descriptions are actually bound to the variable `<var>`. For the quantifiers, we follow the *Repeatability* criterion of [9] (p. 35), which states:

> Any syntactic relation must be either unlimitedly repeatable or non-repeatable.

The criterion marks the difference between complements, which must occur exactly once, and adjuncts, which can occur any number of times, including not at all, motivating the following quantifiers:

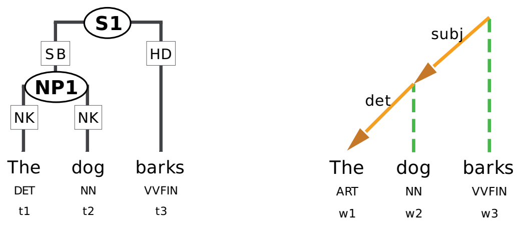
- **first**: Applies the actions only to the leftmost node that matches the criteria. The actions are mandatory; if they cannot be applied, the conversion fails immediately.
- **last**: This quantifier is analog to **first**, but matches the right- instead of the leftmost node, which is useful for right-headed phrases.
- **first?**, **last?**: These two quantifiers behave like their non-?-suffixed versions, except that they do not enforce application.
- **each**: Applies the actions to any node matching the criteria.

3.3.2 Actions

While the quantifiers are used to identify nodes, actions are used to describe the structure which is created. Examples for the two most important actions, head marking and dependency edge insertion, are shown in example 3. The quantified actions are applied from top to bottom.

(3)
```
context { #s:[cat="S"] } {
    first #hd { HD [FREC] } => root #hd;
    first #s { SB [FREC] } => <root> subj #s;
}
context { #np:[cat="NP"] } {
    last #n { NK [pos="NN"] } => root #n;
    each #d { NK [pos="ART"] } => <root> det #d;
}
```

The `root #hd` action marks its node arguments as the head of the current nonterminal, in this case the first (and only the first) occurrence of a node with the edge label `HD`. In the second action, we introduce a dependency between the head and the first node with the edge label `SB`, using the edge insertion action `<root> subj #s`. Since there is only one head per nonterminal—marking more than one node as a head is a conversion error and results in immediate failure—we can refer to it using the implicit variable `<root>`. Since the conversion process is not tied to any



(a) Original (b) Conversion

Figure 1: Conversion Example

specific dependency grammar, the set of edges is not limited, and any alphanumeric string is a valid edge label.

Each node marked as head and each node on the right-hand side of an edge insertion action are marked as *consumed*. In a bound rule, each node is consumed at least once, which means that it cannot be matched in subsequent quantifiers. Consumption is also used to check the completeness of the conversion process after no more rules can be applied.

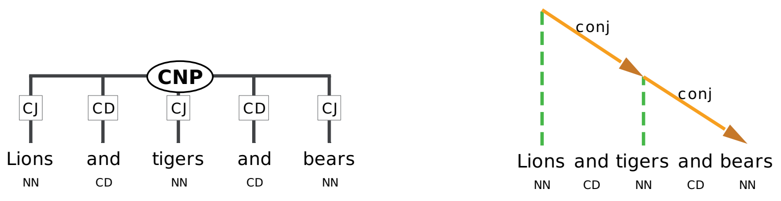
In some cases, it is necessary to drop edges that are present in the constituent tree, like punctuation tokens, which are attached to the virtual root phrase in the NEGRA annotation scheme [3]. The `ignore` action marks a node variable as consumed without introducing a dependency edge.

3.3.3 The Conversion Process

Each nonterminal p, unless all its children are dropped from the structure, must have a unique root head c_p, otherwise the conversion will fail. The head assignments of the rule applications define a surjective mapping of nonterminals to terminals, which are mapped to words in the dependency structure. As an example, we show the steps for converting the structure shown in figure 1a to the dependency graph in 1b, based on the conversion rules in example 3.

1. **Identify the heads**
   $\mathtt{NP}_1 \equiv \mathtt{t}_2 \wedge \mathtt{S}_1 \equiv \mathtt{t}_3$
2. **Create dependency edges**
   $\mathtt{t}_2 \xrightarrow{\mathtt{det}} \mathtt{t}_1 \wedge \mathtt{t}_3 \xrightarrow{\mathtt{subj}} \mathtt{NP}_1$
3. **Replace all nonterminals with an equivalent terminal**
   $\mathtt{t}_2 \xrightarrow{\mathtt{det}} \mathtt{t}_1 \wedge \mathtt{t}_3 \xrightarrow{\mathtt{subj}} \mathtt{t}_2$
4. **Replace constituent tree terminals by words**
   $\mathrm{w}_2 \xrightarrow{\mathrm{det}} \mathrm{w}_1 \wedge \mathrm{w}_3 \xrightarrow{\mathrm{subj}} \mathrm{w}_2$

In the first step, we assign a head to each nonterminal phrase, for instance “dog” (t_2) for NP_1. In step 2, all remaining nodes are connected to their heads using dependency edges. In step 3, all nonterminals in the dependency edge definitions are replaced by equivalent terminals, which are determined using the assignments



(a) Original (b) Conversion

Figure 2: Handling of Coordination

from step 1. The last step replaces all terminals by the equivalent words in the dependency graph.

3.4 Free Rules

Free rules allow for conversion of arbitrarily large structures like coordination phrases. Consider the tree in figure 2a; instead of selecting one conjunct as the head and connecting all other conjuncts to it, it should also be possible to create a chained structure as seen in figure 2b. Since there is no limit to the number of conjuncts, we combine a bound rule and a free rule that invokes itself recursively, as shown in example 4.

(4)
```
rule coord() {
    first #first { CJ [FREC] } => root #first;
    first? #e { CJ [FREC] } => <root> conj coord();
}
context { #cnp:[cat=("CAC"|...|"CVZ")] } {
    each #cd { CD [FREC] } => ignore #cd;
    root coord();
}
```

In the bound rule, any conjunctions (edge label CD) are dropped from the phrase and the free rule `coord` is invoked unconditionally. Like bound rules, free rules have at most one node that is marked as a head, again using the `root` action, which is also used as the final argument to the action in which the free rule is invoked. Therefore, the head of the first invocation of `coord` also becomes the head of the whole coordination phrase.

In `coord`, the leftmost conjunct (edge label CJ) is marked as the head. The second action, quantified with `first?` is invoked only if another conjunct remains, which is connected to the head with the dependency `conj`. Otherwise, the base case is reached and recursion stops.

Free rules are also useful to avoid rule duplication when certain constituent phrases have a similar structure.

Result	Count	Percentage
Complete	43,299	85.79%
Partial[1]	5,902	11.69%
Failure	1,269	2.51%
Total	50,470	100.00%

Table 1: Conversion Coverage in the TIGER Treebank Release 2.1

3.5 Match Cascades

Some phrases have a varying structure. The head of an NP can be a noun, a personal or demonstrative pronoun or a proper name, which are mutually exclusive, but in some cases it is necessary to match some with higher priority than others. So far, this can only be approximated by using a series of quantifiers as shown in example 5. To solve this problem, we introduce cascaded descriptions as shown in example 6.

(5)
```
first? #h { [pos="NK"] } => root #h;
first  #h { [pos="FM"] } => root #h;
```

(6)
```
first #h { [pos="NK"], [pos="FM"] } => root #h;
```

The descriptions in example 6 are matched against the available nodes in the order they are written and behave similar to the rules in ex. 5, but only one head is assigned.

4 Conversion of the TIGER Treebank

As a first experiment, we wrote a rule set for converting the German TIGER treebank [2] into a dependency format. The rule set consists of 14 bound and 5 free rules, with little more than 100 individual actions. Coverage is shown in table 1, which is based on a version of the converter that handles secondary edges transparently. Any secondary edge in the constituent graph is treated like a dominance edge, but is eventually inserted as a secondary dependency. If deactivated, the failure rate raises to 3.51%.

We used the interactive character of our tool to constantly check and improve performance and coverage. We did not perform an exhaustive evaluation, but only errors remain.

5 Conclusion & Future Work

In this paper, we created a declarative formalism for converting constituent trees into dependency graphs and showed that the formalism is already powerful enough

[1]Conversion process results in several unconnected substructures.

to convert more than 97% of the TIGER treebank. Rule writing and debugging is supported by the graphical interface which converts on the fly and delivers error and coverage messages.

We implemented the formalism in *ladon*[2], a library for structured linguistic annotation data. We also developed a new tool for browsing and searching treebanks that supports interactive writing of conversion rule sets as well as transparent conversion and searching.

5.1 Converting the Penn Treebank

Following the work by [9] and [14], we will create a rule set for the Penn Treebank. In the current implementation, the context for bound rules is very limited, since only local features of the node can be used. In some cases, it is desirable to choose conversion rules based on a larger context, and to be able to add further sources of information. Local conversions, for example in [12] or extensions of it like in [13] can run into linguistic problems and inconsistencies, which are largely due to the Penn Treebank annotation scheme. For example, the distinction between objects, adjuncts and indirect objects is not always possible, the distinction between appositions and conjunctions needs non-local context, and the distinction between PP complements and adjuncts is partly underspecified in the Penn Treebank.

References

[1] Adriane Boyd, Markus Dickinson, and Detmar Meurers. On Representing Dependency Relations – Insights from Converting the German TiGerDB. In *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT 2007)*, pages 31–42, Bergen, Norway, 2007.

[2] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories, TLT 2002*, Sozopol, Bulgaria, 2002.

[3] Thorsten Brants, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut, and Hans Uszkoreit. Das NEGRA-Annotationsschema. Technical report, Universität des Saarlandes, Saarbrücken, Germany, 1997.

[4] Michael A. Covington. GB theory as Dependency Grammar. Technical Report AI1992-03, University of Georgia, Athens, Georgia, 1992.

[5] Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation*, Manchester, UK, 2008.

[2]http://www.cl.uzh.ch/kitt/hg

[6] Martin Forst, Nria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, and Valia Kordoni. Towards a dependency-based gold standard for German parsers – The TiGer Dependency Bank. In *Proceedings of LINC-04*, Geneva, Switzerland, 2004.

[7] Katri Haverinen, Filip Ginter, Sampo Pyysalo, and Tapio Salakoski. Accurate conversion of dependency parses: targeting the Stanford scheme. In *Proceedings of Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008)*, Turku, Finland, 2008.

[8] Reiko Heckel. Graph Transformation in a Nutshell. In *Proceedings of the School on Foundations of Visual Modelling Techniques (FoVMT 2004) of the SegraVis Research Training Network*, volume 148, pages 187–198. Elsevier, 2006.

[9] Richard Johansson. *Dependency-based Semantic Analysis of Natural-language Text*. PhD thesis, Department of Computer Science, Lund University, Lund, Sweden, 2008.

[10] Wolfgang Lezius. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora*. PhD thesis, IMS, University of Stuttgart, Stuttgart, Germany, December 2002.

[11] David Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, Boston, MA, 1995.

[12] Joakim Nivre. *Inductive Dependency Parsing*. Springer, 2006.

[13] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, 2007.

[14] Gerold Schneider. Extracting and Using Trace-Free Functional Dependencies from the Penn Treebank to Reduce Parsing Complexity. In *Proceedings of Treebanks and Linguistic Theories (TLT)*, Vaxjö, Sweden, 2003. University Press.

[15] Gerold Schneider. *Hybrid long-distance functional dependency parsing*. PhD thesis, University of Zürich, 2008.

Selectional Preferences from a Latin Treebank

Barbara McGillivray
University of Pisa
E-mail: `b.mcgillivray@ling.unipi.it`

Abstract

We present a system for automatically acquiring selectional preferences for Latin verbs. We use the *Index Thomisticus* Treebank Valency Lexicon and an enriched version of Latin WordNet as the reference conceptual hierarchy.

1 Introduction

The linguistic community today can rely on large annotated corpora, lexical resources and Natural Language Processing (NLP) tools for several modern languages. Compared with these, Latin is a low resource language. In the past years various projects have started aiming at filling this lacuna. Three treebank projects are ongoing, sharing annotation style: Latin Dependency Treebank (LDT), *Index Thomisticus* Treebank (IT-TB) and PROIEL Treebank.[1] Among the lexical databases, Latin WordNet (LWN) [6] consists of around 10,000 Latin lemmas mapped into the parallel structure of MultiWordNet [3]. The syntactic content of treebanks joined with the semantic information from LWN allows for further research in distributional computational semantics and in several NLP applications: word sense disambiguation, anaphora resolution, parsing, etc.

This paper deals with the issue of automatically extracting semantic information from syntactically annotated Latin corpora. In particular, we aim at extracting the verbs' selectional preferences (SPs), i. e. the semantic preferences of verbs on their arguments. For example, the subject position of the transitive verb *think* is usually filled by lexical items whose semantic properties include being human. The background for this work is the IT-TB Valency Lexicon [5], which collects syntactic arguments of verbs occurring in the IT-TB and represents them in a structured, easily searchable way. It is automatically extracted from this treebank and updated as the annotation proceeds. Following the method illustrated in [1] developed for a cognitive model, we exploit the semantic hierarchy of LWN to map lexical items into concepts and extract their semantic relations among them. Then, SPs are calculated as probability distributions over these semantic features. The result is a rich computational resource for the variety of Latin attested in the IT-TB.

[1]The sizes of these treebanks range between 50,000 words (LDT) and 100,000 (PROIEL). See [5] for references.

2 Background and new contribution

Over the last decade, research into automatic acquisition of SPs from corpora led to several systems which rely on supervised and unsupervised methods. Given a set of argument headwords (for example, *doctor*, *child* as subjects of *think*), these approaches perform a generalization step over unseen cases (*human being*). The generalization problem is represented in WordNet (WN) approaches in terms of preference probabilities over a noun hierarchy, and the goal is to find the appropriate noun classes for each verb-argument pair (the probability of *human being* appearing in the subject position of *think*). This is achieved using statistical tools from information theory, statistical testing and modeling (see [4] for references).

All these models start from single verb-argument occurrences collected from large corpora to infer the probability that the verb's argument position is filled by a class of nouns. Since our dataset is extracted from a small treebank (63,000 tokens), a large number of low frequency verb-noun distributions are observed. Nevertheless, these low frequencies are not necessarily a property of the Latin verbs: they follow from a small which underestimates variance. This can be remedied by grouping the observations into larger clusters. For these reasons, the method illustrated in [1] proved effective when dealing with the peculiarities of our data.

Although we work on the same language and with treebanks comparable in size, our system also differs from the one described in [2]. Bamman and Crane report experiments on extracting SPs from a 3.5 million word Latin corpus which was automatically tagged and parsed using the LDT as training set. The large size allows for better variance estimates and more representative frequencies: this overcomes the problems caused by noisy parsed data. SPs are then extracted through the log likelihood test, a technique also used in collocation extraction. The output consists of association scores between single verbs and single nouns occurring as their arguments. Given the high complexity of the LDT (various authors, genres and time periods), their system makes finer-grained distinctions between the specific usages in different authors, eras, and genres.

While having the same annotation style, IT-TB and LDT differ as to their composition: IT-TB contains works by one author (Thomas Aquinas), belonging into one genre (philosophy). Hence, the variability of the lexicon in the IT-TB is not as extreme as in the LDT, allowing us to treat the corpus as a homogeneous whole. This also implies that a higher number of verb (and noun) instances are typically found that share the same sense (i. e. WN synset). This decreases the probability of finding very low frequency associations between a verb sense and a noun sense, and partially improves the accuracy of the extraction system. Moreover, instead of finding association scores between verbs and nouns in an argument position, we aim at calculating the probability of a WN concept occurring in an argument position for a given verb. This proves to be effective in a lexicographic perspective, where broader semantic classes rather than single words are required.

3 Acquisition of selectional preferences

Alishahi and Stevenson [1] propose a computational model for SP induction in a cognitive framework. In their model, each verb usage is a *frame*, that is a collection of syntactic and semantic features, such as the number of verbal arguments, the syntactic pattern, and the semantic properties of each argument. By semantic properties they refer to the lists of WN hypernym synsets for each word. A *construction* is defined as a collection of frames probabilistically sharing some feature values, for example the transitive construction: a construction clusters frames together based on their syntactic and semantic features. For each (verb, argument position) pair, a probability distribution over a set of semantic properties is calculated; this probability distribution represents the verb's SPs.

We adapted the definitions of frame, syntactic and semantic features to the data at hand. In the IT-TB Valency Lexicon each verbal form occurring in the IT-TB corresponds to a lexical entry recording syntactic, morphological and lexical information on the verb's arguments. For example, the sentence[2]

(1) dominus discipulis formam baptizandi dedit.
Lord-NOM.M.SG disciples-DAT.M.PL form-ACC.F.SG baptize-GERUND-GEN give-PRF.3SG
"the Lord gave to the disciples the form of the baptism."

represents a frame for an active ditransitive occurrence of the verb *do* ("give"); it is recorded in the lexicon as the following subcategorization structure (SCS):[3]

$$do + A\_Sb[nom]\{dominus\}, Obj[acc]\{forma\}, Obj[dat]\{discipulus\} \qquad (1)$$

The argument positions (or slots) for the active form of *do* are a nominative subject (A_Sb[nom]), an accusative object (A_Obj[acc]) and a dative object (A_Obj[dat]).[4] The lemmas, or *fillers*, of the lexical items occurring in these positions are: *dominus* ('Lord'), *forma* ('form'), and *discipulus* ('disciple'). We assigned the SCS structures to the set of syntactic features of the frame ($feature_1$).

In order to define the semantic features of each frame, we referred to the LWN database, which contains around 10,000 lemmas aligned with the English WN. The mapping links each Latin synset to an English synset and defines a "lexical gap" when this is not possible. Since the coverage of this resource is low with respect to our data,[5] we semi-automatically added new Latin lemmas to the hierarchy in the following way. For each lemma L (for example *abiectio*), we collected its Italian and/or English translations T (for example 'avvilimento', 'abbattimento', 'dejection', 'despondency') by using electronic versions of Latin-to-Italian and Latin-to-English dictionaries. Then, we selected the synsets of T that are relevant to the

[2] Thomas, *Super Sententiis Petri Lombardi*, IV, Distinctio 8, Quaestio 1, Articulus 3C, Argumentum 2, 3-3.4-1.

[3] 'A' stands for 'Active', 'Sb' for 'subject', 'Obj' for 'object', 'nom' for 'nominative', 'acc' for 'accusative', and 'dat' for 'dative'.

[4] The linear order of these elements in the sentence is not recorded: this choice is due to the relatively free word order in Latin sentences.

[5] 1027 fillers out of 2934 and 90 verbs out of 559 were not present in the lexical database.

senses of L; thanks to the alignment in MultiWordNet, we finally assigned L to the Latin synsets corresponding to these selected senses of T, if any (*humiliatio-humilitas-indignitas*; *contritio*; *demissio*).[6]

For each argument position, the semantic properties of a frame ($feature_2$) are the set of WN hypernyms of the fillers for that slot. For example, the semantic properties for the slot $A\_Sb[nom]$ in (1) are all the hypernyms of *dominus*.

Finally, the semantic properties of the verb belonging to a frame are the list of its WN synsets ($feature_3$): in (1) they coincide with the synsets of *do*.

3.1 Bayesian clustering of frames

In the approach suggested by [1], a frame is clustered into a new construction according to the probabilistic similarity between its features and the features of the frames already included in the construction. This way constructions are created incrementally by means of a Bayesian process. A construction K is chosen for a frame F if it maximizes the probability $P(k|F)$ over all constructions k (including a new construction), that is (after Bayes' theorem) if it maximizes the product $P(k)P(F|k)$. We set the prior probability $P(k)$ to the number of frames contained in k divided by the total number of frames. If we assume that the frame features are independent, $P(F|k)$ is the product of $P_i(feature_i(F)|k)$ for i=1,2,3: $P_i(feature_i(F)|k)$ is the probability that the i^{th} feature displays in k the value it has in F, that is $feature_i(F)$.

$Feature_1$: for the syntactic properties, we estimated $P(feature_1(F)|k)$ by using the following maximum likelihood formula:

$$P(feature_1(F)|k) = \frac{\sum_{h\in k} synt\_score(h,F)}{n_k}$$

where $synt\_score(h,F) = \frac{|SCS(h)\cap SCS(F)|}{|SCS(F)|}$ (*syntactic score*) is the number of syntactic slots shared by h and F over the number of slots in F. This accounts for the degree by which two frames h and F differ in their syntactic patterns (SCSs).[7] For example, let F be the frame given by the verb *coniungo* ('join') + $P\_Obj[dat]\{finis\}, P\_Sb[nom]\{calor\}$ and h be the frame *adiungo* ('join') + $P\_Obj[dat]\{terminus\}$. The algorithm clustered F into a construction containing h: hence, $synt\_score(h,F) = \frac{1}{2}$.

$Feature_2$: for each argument position a in F, $P(feature_2(F)|k)$ is

$$P(feature_2(F)|k) = \frac{\sum_{h\in k} sem\_score_a(h,F)}{n_k} \qquad (2)$$

[6] This way, we were able to add 401 new noun lemmas + 90 verb lemmas to 2056 already existing Latin synsets.

[7] Given the high frequency of omitted arguments in Latin sentences, the chances of an exact match between the two SCSs are low. For this reason, we did not define the syntactic score as a binary function.

where $sem\_score_a(h,F) = \frac{|S(h) \cap S(F)|}{|S(h) \cup S(F)|}$ (*semantic score*) accounts for the degree of overlap between the semantic properties $S(h)$ of h and the semantic properties $S(F)$ of F (for argument a). In the previous example, *terminus* ('limit') and *finis* ('end') are the lexical fillers of the $P\_Obj[dat]$ slot for h and F, respectively. The semantic score is $\frac{1}{5}$ because the intersection between the semantic properties of the two words contains one item (*abstraction*), as shown below:[8]

terminus: indefinite_quantity, mensura-modus-quantitas ('measure-quantity-amount-quantum'), abstraction.
finis: locus-punctum ('point'), aetas-circumductio-circumductum-continuatio-periodus-sententia--spatium ('time_period- period-period_of_time-amount_of_time'), abstraction.

*Feature*3: the probability of displaying in k the value that F has *feature*$_3$ is

$$P(feature_3(F)|k) = \frac{\sum_{h \in k} syns\_score(h,F)}{n_k} \tag{3}$$

where $syns\_score(h,F) = \frac{|Synsets(verb(h)) \cap Synsets(verb(F))|}{|Synsets(verb(F))|}$ (*synset score*) calculates the degree of overlap between the synsets for the verb in h and the synsets for the verb in F over the number of synsets for the verb in F; n_k is the number of frames in k.

The algorithm uses smoothed versions of all the previous formulas and clusters a frame into a construction after taking into account its syntactic features (relative to the subcategorization pattern) and its semantic properties (relative to both the verb and the lexical fillers of the verb's arguments).

3.2 Selectional preferences

The clustering allows us to perform the generalization step over unseen cases while predicting the probability that a noun n is an argument filler for a verb v in an argument position a; this probability is calculated as the sum of $P_a(n,k|v)$ over all constructions k and can be approximated as the product $P(k,v)P_a(n|k,v)$. $P(k,v)$ is the probability that v occurs in construction k and is estimated as the smoothed relative frequency of v occurring in k. On the other hand, we calculate $P_a(n|k,v)$ as

$$P_a(n|k,v) = \frac{\sum_{h \in k} sem\_score(h,n) \cdot syns\_score(h,v)}{n_{k_v}}$$

where $sem\_score(h,n)$ is the semantic score between the set of semantic properties of the fillers for a in h and the set of semantic properties of n (see formula (2)); $syns\_score(h,v)$ is the synset score between the synsets of v and the synsets of the verb in h; finally n_{k_v} is the number of frames contained in k whose verbs share with v the same synset. Note that frames containing verbs semantically similar but not identical to v do contribute to the probability, thus contributing with an innovation with respect to Alishahi & Stevenson's system. This is particularly important when dealing with few occurrences of v that would not allow further generalization over unseen cases; as noted previously, this is a frequent case in our dataset.

[8]For reasons of space we only displayed the set of semantic properties of one sense for each word.

4 Conclusion

We propose a new system for automatically acquiring SP which integrates frequencies from a Latin treebank with a translation-analogy-enriched version of LWN. Since this research employs a treebank for a less resourced language, it gave us the opportunity to discuss issues related to the size of treebanks for these languages and the integration with other lexical resources, such as wordnets. We showed how methods developed in computational semantics for extant languages and large corpora can be adapted to the special case of a dead language in order to improve the state of the art of its resources. In particular, our approach deals with low frequency items in a novel way by means of a clustering technique which expands the set of seen occurrences that participate in the generalization step, while calculating selectional preferences. In the near future we plan to evaluate this system both against traditional resources such as dictionaries and thesauri, and against corpus data from other sources. Finally, thanks to the shared annotation, running our system on the LDT and the PROIEL treebank would lead to diachronic investigations on Latin syntax and semantics, while at the same time being a computational challenge.

References

[1] A. Alishahi and S. Stevenson. A cognitive model for the representation and acquisition of verb selectional preferences. In *Proceedings of the ACL Workshop on Cognitive Aspects of Computational Language Acquisition. Prague*, pages 41–48, 2007.

[2] D. Bamman and G. Crane. Building a dynamic lexicon from a digital library. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2008). Pittsburgh*, 2008.

[3] L. Bentivogli, P. Forner, and and Pianta E. Magnini, B. Revising wordnet domains hierarchy: Semantics, coverage, and balancing. In *Proceedings of COLING Workshop on Multilingual Linguistic Resources*, pages 101–108, Geneva, 2004.

[4] C. Brockmann and M. Lapata. Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of the tenth conference on European chapter of the ACL*, volume 1, Budapest, 2003.

[5] B. McGillivray and M. Passarotti. The development of the *Index Thomisticus* Treebank Valency Lexicon. In *Proceedings of the Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH-SHELT&R 2009). Athens*, pages 33–40, 2009.

[6] S. Minozzi. The Latin Wordnet project. In P. Anreiter and M. Kienpointner, editors, *Proceedings of the 15th International Colloquium on Latin Linguistics (ICLL)*, 2009.

Annotation Quality Checking and Its Implications for Design of Treebank (in Building the Prague Czech-English Dependency Treebank)

Marie Mikulová and Jan Štěpánek

Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
E-mail: {mikulova,stepanek}@ufal.mff.cuni.cz

Abstract

This article presents the system for annotation quality checking, proposed and used during the building of the Czech part of the Prague Czech-English Dependency Treebank. At first, the treebank project is introduced, as well as its basic principles and annotation process. The second part of the article pursues in detail one of the important phases of the annotation process, namely how the correctness of the annotated data is automatically and continuously checked during the process. The system of annotation quality checking is demonstrated on several particular checking procedures concerning syntactical phenomena. We try to evaluate the contribution of the system not only to the quality of the data and annotation, but also to the corpus design, impact on annotation rules and the annotation process as a whole.

1 Introduction

The annotation of a corpus is always a complex task, especially if the corpus is large and the added linguistic information is rich. The organization and management of the annotation process needs to be elaborate. If the linguistic information attached to input data (sentences) is rich and complex, the common method, multiple parallel annotation and measurement of inter-annotator agreement, becomes too costly and inapplicable for capacitive reasons. For the Prague Czech-English Dependency Treebank (PCEDT), an example of such a large complex corpus, a system of automatic quality checking of the annotated data was developed to avoid multiple parallel annotation (however, subsets of the data were still annotated in parallel).

2 The Prague Czech-English Dependency Treebank

2.1 Basic Principles

The PCEDT is planned to be a corpus of (deeply) syntactically annotated parallel texts (in English and Czech) intended chiefly for machine translation experiments. The texts for the PCEDT (its first version was described in (Čmejrek et al., 2004)) were taken from the Penn Treebank (Marcus et al., 1993), which means there are mostly economical articles from the Wall Street Journal. 2312 documents were used in the PCEDT (approximately 49,000 sentences) that are manually annotated using constituent trees in the Penn Treebank. For the Czech part of the PCEDT, the English texts were translated into Czech.

As a base of the process of creation of the corpus (hierarchical system of annotation layers, annotation rules) we use the already finished Prague Dependency Treebank (PDT) 2.0 (Hajič et al., 2006). While organizing the annotation of the PCEDT (especially its Czech part, which is the main concern of this article), we will prop ourselves upon multifarious experiences (both positive and negative) gained from the production of the PDT 2.0.

Similarly to the PDT 2.0, written sentences in the PCEDT are represented on three layers: morphological layer (lemmas, tags, morphological categories), analytical layer (surface structure, dependencies, analytical functions) and tectogrammatical layer.

The tectogrammatical layer contains all the information that is encoded in the structure of the sentence and its lexical items: the deep, semantic-syntactic structure, the functions of its parts, values of the "deep" grammatical categories, the coreference and the topic-focus articulation including the deep word order. Every sentence is represented by a tectogrammatical tree. A node of the tree either represents a semantic unit present on the surface shape of the sentence (an autosemantic word with its function words like prepositions, subordinating conjunctions, auxiliary verbs converted into various node attributes called "grammatemes") or it is a newly established node that has no counterpart on the surface—in case of ellipsis.

We adhere to the stand-off annotation principle: the layers of annotation are separated from the input data and from one another, they are interlinked by references leading always from the hierarchically higher layers to the lower ones. For example, there are two references to the analytical layer from a tectogrammatical node representing a prepositional group (as one syntactic-semantic unit): one pointing at the preposition and one at the noun.

2.2 Annotation Procedure

Tectogrammatical (deep syntactic) layer of the PCEDT that is currently being built (but also of the already accomplished PDT 2.0) represents a really broad linguistic annotation. There are 39 different attributes, for a node of a tectogrammatical tree in the PDT 2.0 there are 8.42 attributes filled on average. The annotation manual

(Mikulová et al., 2006) has more than 1000 pages.

In case of such an extensive annotation project, we find the following three aspects necessary:

- division of the annotation into several phases
- periodic measurement of inter-annotator agreement
- continuous checking of the annotation correctness

Although the annotators get their data automatically preprocessed, so that they do not have to build the syntactical tree from the very beginning (Klimeš, 2006), the annotation of the tectogrammatical layer in phases is unavoidable, because no annotator is able to keep all the annotation rules for all the annotated phenomena in her head. Moreover, the more information the annotator has to attach to the data, the more likely he or she omits some of the details.

Therefore, for the tectogrammatical annotation, we plan for the following phases:

1. building a tree structure, dealing with ellipsis included; assignment of functors and valency frames, links to lower layers (10 attributes).
2. annotation of subfunctors (fine grained classification of functors, 1 attribute).
3. annotation of coreference (4 attributes).
4. annotation of topic-focus articulation, rhematizers and deep word order (3 attributes).
5. annotation of grammatemes, final form of tectogrammatical lemmata (17 attributes).
6. annotation of remaining phenomena (quotation, named entities etc.)

The first phase is the most difficult, each annotator is responsible for the whole structure of the tree and correct values of ten attributes. All these attributes are connected with the structure and deep syntactical functions of nodes. Annotators do not have to pay attention to anything else.

To determine the amount of annotated information that does not harm the quality of the data is obviously difficult. Our belief that the current schedule of phases constitutes a reasonable and manageable rate seems to be justified by the measuring of inter-annotator agreement. The quality of the data is regularly guarded by the system of automatic checking procedures (see the following section).

3 System for Annotation Quality Checking

3.1 Motivation

A higher quality of the annotated corpus data is usually attained by parallel annotation of the same data. Particular annotated data from different annotators are mutually compared, the differences are found and manually corrected. Such a method is very expensive (with regard to time and work) and becomes impossible if the treebank is large and the annotated information is complex. In one hour, one annotator can annotate 9.2 sentences in average in the first phase of the tectogrammatical annotation of the PCEDT. Annotation of the whole treebank (about 49,000 sentences) by one person would therefore take 5326 hours. If a person worked regularly for 20 hours a week (half-time job), the whole treebank in its first phase would take 5 years. It is clear that if we want to keep the volume of the annotated information (i.e. the annotation rate), we have to search for a different way to guard the quality of the annotated data than the usual parallel annotation.

Consequently, already during the building of the PDT 2.0, a system for automatic checking of the data was developed. The real checking took place when all the annotation had finished. The checking and fixing phase was quite complex and time-consuming; moreover, in some cases, the changes were not realized full-scale (Štěpánek, 2006). We want to avoid such a procedure in the development of PCEDT. The system of automatic checking procedures is now fully integrated into the annotation process.

3.2 Design of the Automatic Checking Procedures

Checking of the annotated data takes place continuously during the annotation process. At the beginning of the process, many automatic checking procedures were proposed and created in accordance with the annotation guidelines and more are added even during the process. Contrary to (Boyd et al., 2007) or (Dickinson, 2005), all the procedures were programmed manually, even though some of them were based on generalization of the output of an automatic inconsistency lookup.

The checking procedure proposals are based on the fact that many annotation rules imply that particular phenomena cannot (or have to) occur in the annotation output. They mainly combine attribute values and the structure of a tree.

For example, a simple check (coord002) states that every coordination has at least two members and reports all one-member (and zero-member) coordinations as errors. Another check (structure001.2) states that the root of a tectogrammatical tree has only a limited set of possible functors (PRED for a predicate, DENOM for nominative clause, PARTL for interjection clause, VOCAT for vocative clause). There is also a converse check (structure001.1) monitoring that no dependent node has the PRED or DENOM functor. The checks structure021.1 and structure021.2 guard the fact that some types of nodes, i.e. the nodes added to a tree in case of ellipsis of the governing node (such a node either has a special tectogrammatical lemma #EmpVerb or #EmpNoun or is a copy of a different node in the tree) can by definition never be leaves of a tree.

Each checking procedure has the same structure: its name consists of a category (see below) and number. A short comment explains the purpose of the procedure. All the procedures are written in Perl, as well as the main annotation tool TrEd (Pajas and Štěpánek, 2008). The output of each procedure is a table whose first column contains the name of the procedure followed by a short explanation of

```
#!btred -N -T -t PML_T -e coord()

package PML_T;

$NAME='coord002';

## Every coordination has at least two members.

sub coord {
  writeln("$NAME\tmembers\t".ThisAddress($this))
    if IsCoord($this)
       and scalar(grep $_->{is_member},$this->children) < 2;
} # coord
```

Figure 1: Checking procedure coord002

the violation of the monitored invariant. There can be any number of additional columns with any information, but the last column must contain the address of the problematic node. The annotation tool TrEd can open files with lists of addresses and can subsequently open a given tree with the cursor on a given node, and after a correction is made, it can switch to the next address in the list. An example of a checking procedure is shown in Figure 1.

The checking procedures should be accurate, i.e. they should report an error only if the data are not correct and should not report it if not. To achieve this, exceptions might be added to some of the procedures (stating something like “do not report error if the identifier of the node is T2431, the node is a leaf and its parent has functor ACT”). Several procedures had to be abandoned because of the number of exceptions getting too high.

By its name and function, each checking procedure can be classified into one of five categories:

(1) structure This category constitutes the fundamental part of the checking procedures. The procedures in this category check the structure of tectogrammatical trees, i.e. the relations between governing and dependent nodes. Particular types of nodes presume by definition only particular types of governing and/or dependent nodes. Any departure from these invariants is reported by the procedures. Besides those already mentioned, another typical example is structure027.1: it states that adjectival attribute (functor RSTR) never depends on a verb; or structure016.1, that assures that every negated verb has a #Neg child.

(2) coord The checking procedures guarding the correctness of coordination structures form a separate group. An example of the procedure coord002 was already given in Figure 1. Other procedures in this category follow from the rules stating that some types of functors can never be coordinated together, e.g. an inner participant (argument) can be in coordination only with an argument of the same sort (coord004.4).

(3) valency The next group of checking procedures is related to valency. During the tectogrammatical annotation, each verb is assigned a valency frame from the valency lexicon describing the verb's meaning and laying down which valency modifications pertain to the given verb. (Procedure valency001.1 checks that the valency frame is assigned where required). If some of the obligatory modifications are not present on the surface, they have to be added to the tectogrammatical tree in form of so called newly established nodes. The valency checking procedures guard that all those modifications were correctly added, mainly, whether none of them is missing (valency003.2), that no modification is redundant (valency003.3), and also that the form and auxiliary words used to express the modification are listed as possible in the lexicon (valency003.4). Another checking procedure (valency004.1) monitors whether all copied verb nodes have the same valency frame assigned as their original (as in the ellipsis *John loves Mary and Peter [loves] Jane*).

(4) links The checking procedures in this category check the correctness of the links from the tectogrammatical nodes to the analytical nodes. For example, for every analytical node representing a word (i.e. not punctuation) there must be a link from a tectogrammatical tree (links001.1.1). However, the same analytical node can be linked as an auxiliary word from different tectogrammatical nodes only if the tectogrammatical nodes are coordinated in one coordination, or they or their parents have the same tectogrammatical lemma, or if all but the first tectogrammatical node are predicates with PREC children (a particle refering to a preceding context). Other procedures are based on the fact that some types of tectogrammatical nodes anticipate particular types of links, or some types of links are forbidden for them. For example, there can be no link to a preposition from a tectogrammatical node with the functor DENOM, representing a nominative clause and the same holds for a node with the functor VOCAT representing vocative clause (links003.1).

(5) attribute Checking procedures in this category do not care about the structure of a tree, only attributes of a single node are checked. All the procedures created so far focus on the tectogrammatical lemma (no other non-structural attribute is annotated in the first phase of the process). For example, attribute001.1 checks that reasons are given for every change in pre-generated tectogrammatical lemma.

Currently, there are 50 checking procedures with 103 possible violations reported (55 in the category structure, 11 coord, 15 valency, 19 links and 3 attribute).

3.3 Corrections

The checking procedures return a list of erroneous (questionable) positions in the data. The annotator gets his or her data back for corrections, manually fixing each position. As was already mentioned in the previous section, the annotation tool TrEd and the output of the procedures are compatible so that annotators only see the problematic sentences and nodes.

The checking procedures are run periodically after a given volume of the data has been annotated by an annotator (1000 sentences) or once a quarter. An error in the annotation is often reported by more than one checking procedure, each of them can report a different node affected by the error. This would confuse annotators: once they would have corrected an error in a list from one procedure, they would encounter the corrected sentence again in a different list. Moreover, by rearranging nodes in a tree an address might point to a different node than before. Therefore, the list is always filtered to contain each sentence just once in one run.

All the data (the already corrected ones included) are checked every time (in case a new check existed), and after one run of corrections, the data are checked again and again while there are any errors left (there might be some errors left by the filtering and new errors can arise in fixing the old ones).

3.4 Limits of the Automatic Quality Checking

Automatic checking procedures improve the quality of the data not only by fixing the present errors, but also by providing a feedback to the annotators (because each annotator fixes his or her own data, i.e. his or her own errors) and thus eventually improving the future annotation.

Nevertheless, even the automatic checking procedures have their limits, they cannot be used to check everything. One of the most difficult tasks is the validation of functors. The definitions of functors are very general, there are no invariants and no clues for a procedure to catch on. When assigning a functor, the annotator uses mainly his or her intuition. Of course, there is plenty of phenomena for which no automatic checking is possible. If it were possible to check the whole annotation automatically, it would be as well possible to annotate the whole data with a machine. The point of manual annotation is to obtain the data that cannot be gained automatically, because human intuition and decision of an educated annotator are needed to produce them.

However, we are aware that even the intuition of the annotators has to be revised. It is necessary to monitor whether the intuition of the annotators is roughly the same, i.e. that two different annotators will decide in (roughly) the same way in particular cases.

Consequently, some data files have been picked up for parallel annotation. The differences between annotators that are left after correcting the output of the checking procedures indicate differences in intuition, different approach of the annotators to the phenomena the checking procedures are not able to capture. We measure an inter-annotator agreement, both for individual attributes and overall, on the files annotated in parallel. A low agreement signals that the annotation rules are too loose and we should ponder whether the rules should be made stricter.

4 Implications of the Quality Checking for the Design of the Treebank

Our system for annotation quality checking was primarily developed to search for errors in the annotated data, but its contribution for the treebank building is much broader. The checking procedures not only find errors, but also show problems related to the design of the treebank: reveal vague annotation rules, contribute to the appreciation of the annotators and in effect to the evaluation of the whole annotation procedure.

4.1 Evaluation of Annotators

Using the list of errors generated by the checking procedures, we can count how often the annotators make errors (only those errors the procedures can detect, of course). Long term regular monitoring (since the start of the annotation process) shows that the differences in error rate between annotators can be huge and that all the annotators keep their positions: no one gets markedly better nor worse (so to say, some annotators are simply more thorough than others). The comparison of veteran annotators and the new ones that annotate only for a short time is also interesting: it shows that knack, practice, and experience lead to quality of the annotation.

Table 1 shows the error rate counted on all the data annotated so far (73% of the treebank) for all the annotators individually (the number of errors made by an annotator divided by the number of sentences annotated by her), the total error rate ALL (the number of all errors found divided by the number of all the sentences annotated) and the error rate in the original data ORG (not annotated data, just automatically preprocessed; the number of sentences is lower than the number of annotated sentences because sentences annotated in parallel are counted just once, unlike in ALL).

Annotators can be evaluated by other means, too, for example by measuring the inter-annotator agreement. Such an evaluation is a part of the project as well, but it is not a subject of this article. For more details of the results of the evaluation of the project, see Mikulová and Štěpánek (2009).

4.2 Refining the Annotation Rules

The checking procedures has also lead to a refinement of the annotation rules. A condition of a checking procedure follows from (a combination of) various rules in the annotation manual, therefore one would suppose it holds everywhere in the

Annotator	Errors / Sentences	Errors per Sentence
ma	3 271 / 6 026	= 0.54
al	1 214 / 3 213	= 0.38
iv	2 648 / 8 125	= 0.33
ji	301 / 1 064	= 0.28
mi	430 / 1 786	= 0.24
ka	1 834 / 8 132	= 0.23
le	373 / 1 903	= 0.20
ol	1 177 / 6 828	= 0.17
ALL	12 139 / 39 609	= 0.31
ORG	119 090 / 34 862	= 3.42

Table 1: Number of errors per sentence

data. However, cases that are not errors are often found in the output of some checking procedures (mainly after their first run) among evident mistakes. Those are phenomena that were forgotten when formulating the rules. Thus, as a consequence of application of checking procedures to concrete natural language data, the original annotation guidelines get refined.

For example, the annotation manual states that copied verb has always the same valency frame as the original one (a verb is copied in case of actual or textual ellipses of the governing verb). The checking procedure guarding this rule was already described (valency004.1). However, the real data showed many exceptions to the rule and pointed out some errors in the valency lexicon as well. In case of an actual ellipsis of a governing verb, the manual completely omits the cases of a metaphoric, phraseological or otherwise unusual usage of the verb, as in

Na konflikt nemá dost pozornosti ani [na něj nemá] žaludek.
For a conflict, he does not have enough attention nor [he has] stomach.

"*Have attention*" and "*have stomach*" are two different meanings of the verb *to have*, thus two different valency frames—it is an exception to the rule in the manual, but also for the checking procedure. The procedure indicates inaccuracies in the valency lexicon as well when one meaning is split into several valency frames, for example *Akcie uzavřely na burze smíšeně*, *Akcie uzavřely na burze níže*, and *Akcie uzavřely na burze s mírným poklesem* (*Stocks closed mixed*, *Stocks closed lower*, *Stocks closed down modestly*). In the sentence

Akcie společnosti A uzavřely na burze smíšeně a akcie společnosti B [uzavřely] s mírným poklesem.
Company A's stock closed mixed and company B's [stock closed] down modestly.

the condition of the identity of valency frames cannot be met.

The output of the checking procedures can be compiled into a table showing the type of the most common errors. Among others, they may indicate problems in

Checking Procedure	Occurences	Percentage	Note
valency003_2_PAT_missing	883	7.27	
links001_6.1_same_aux	700	5.77	
valency003_2_ACT_missing	623	5.13	
links001_1.1_no_tnode	438	3.61	
valency001_1_no_frame	405	3.34	
valency003_4_wrong_aux	387	3.19	
structure016_1_no_neg	378	3.11	
attribute001_1_t-lemma	352	2.90	
structure003_1_fphr_lemma	348	2.87	Foreign phrase
valency003_1_invalid_lemma	345	2.84	The same lemma in the data and lexicon

Table 2: Most common types of errors. Procedures without a note are described in section 3.2.

annotation guidelines or the hardest tasks for the annotators. The ten most common errors are shown in Table 2, where the second column presents the number of occurrences and the third column presents the percentage of the given error type among all the errors found.

5 Conclusion

In this paper, we have presented our automatic system for annotation quality checking, which has been proposed and used during building the Czech part of the Prague Czech-English Dependency Treebank. The automatic checking procedures are programmed according to both the explicit annotation rules from the guidelines and the output of an automatic inconsistency lookup, and they monitor the quality of the data already during the annotation process. The output of the procedures is a list of problematic nodes that can be subsequently manually corrected.

The system for annotation quality checking was primarily developed to search for errors in the annotated data, but its contribution is much broader. The checking procedures not only report errors in the annotation, increasing the quality of the data, but also point out problematic annotation rules, contribute to the evaluation of the annotators and of the whole annotation procedure.

Acknowledgement

The research reported in this paper was supported by the LC536, GAUK 22908/2008, and FP7-ICT-2007-3-231720.

References

A. Boyd, M. Dickinson, and D. Meurers. Increasing the recall of corpus annotation error detection. In *Proceedings of TLT 2007*, volume 1, Bergen, Norway, 2007. NEALT Proceedings Series. 3.2

M. Dickinson. *Error detection and correction in annotated corpora*. PhD thesis, The Ohio State University, 2005. 3.2

J. Hajič et al. The Prague Dependency Treebank 2.0. CD-ROM, 2006. URL http://ufal.mff.cuni.cz/pdt2.0. Linguistics Data Consortium Cat. No. LDC2006T01. 2.1

V. Klimeš. *Analytical and Tectogrammatical Analysis of a Natural Language*. PhD thesis, Charles University in Prague, 2006. 2.2

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, (19):313–330, 1993. 2.1

M. Mikulová et al. Annotation on the tectogrammatical level in Prague Dependency Treebank. Annotation manual. Technical Report TR-2006-30, ÚFAL, Prague, 2006. 2.2

M. Mikulová and J. Štěpánek. Annotation procedure in building the Prague Czech-English Dependency Treebank. In *Proceedings of Fifth International Conference SLOVKO*, Bratislava/Smolenice, 25-27 November 2009. In print. 4.1

P. Pajas and J. Štěpánek. Recent advances in a feature-rich framework for treebank annotation. In *The 22nd International Conference on Computational Linguistics - Proceedings of the Conference*, Manchester, 2008. 3.2

M. Čmejrek et al. Prague Czech-English Dependecy Treebank: Syntactically annotated resources for machine translation. In *4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004. 2.1

J. Štěpánek. *Závislostní zachycení větné struktury v anotovaném syntaktickém korpusu (nástroje pro zajištění konsistence dat) [Capturing a Sentence Structure by a Dependency Relation in an Annotated Syntactical Corpus (Tools Guaranteeing Data Consistence)]*. PhD thesis, Charles University in Prague, 2006. 3.1

TEI P5 as an XML Standard for Treebank Encoding[*]

Adam Przepiórkowski

Institute of Computer Science, Polish Academy of Sciences
and Institute of Informatics, University of Warsaw
E-mail: `adamp@ipipan.waw.pl`

Abstract

The aim of the paper is to show that a subset of Text Encoding Initiative Guidelines is a reasonable choice as a standard for stand-off XML encoding of syntactically annotated corpora. The proposed TEI schema — actually employed in the National Corpus of Polish — is compared to other such candidate standards, including TIGER-XML, SynAF and PAULA.

1 Introduction

The need for text encoding standards for language resources (LRs) is widely acknowledged: within the International Standards Organization (ISO) Technical Committee 37 / Subcommittee 4 (TC 37 / SC 4), work in this area has been going on since the early 2000s, and working groups devoted to this issue have been set up in two current pan-European projects, CLARIN (`http://www.clarin.eu.`) and FLaReNet (`http://www.flarenet.eu/`). It is obvious that standards are necessary for the interoperability of tools and for the facilitation of data exchange between projects, but they are also needed within projects, especially where multiple partners and multiple levels of linguistic data are involved.

Given the existence of a number of proposed and *de facto* standards for various levels of linguistic annotation, the starting point of the design of a specific schema to be used in a particular project should be careful examination of those standards: in the interest of interoperability, creation of new schemata should be avoided. But, if a number of standards are applicable, which one should be chosen and on what grounds? And if constructing a schema for a particular linguistic level in a particular project should start with an overview and comparison of existing

[*]The work reported here, carried out within the National Corpus of Polish project funded in 2007–2010 by the research and development grant R17 003 03 from the Polish Ministry of Science and Higher Education, has its origins in a series of papers co-authored by Piotr Bański, dealing with the encoding of lower levels of linguistic information and with the metadata; cf. Bański and Przepiórkowski 2009 and Przepiórkowski and Bański 2009a,b. The paper also profited from insightful comments by TLT8 reviewers.

standards — a time consuming task which may be hard to justify within the limits of a project budget — would it not be easier and cheaper to construct one's own focused schema from scratch?

The aim of this paper is to examine the most popular standards and to make specific recommendations concerning the encoding of syntactic information. We look at four standards: 1) the very specific and commonly used TIGER-XML schema, 2) SynAF, a more general model derived from TIGER-XML, currently under development as an ISO standard, 3) PAULA, a schema for the representation of various linguistic levels, and 4) version P5 of the Text Encoding Initiative (TEI) Guidelines, proposing multitudinous mechanisms for representing multifarious aspects of text encoding. Perhaps because of this richness, TEI is the least obvious candidate for treebank encoding, but it is the one that we would like to argue for here.

2 Standards and Best Practices for Treebank Encoding

In this section we briefly examine three probably most often cited standards and best practices for treebank encoding: TIGER-XML, SynAF (and related ISO proposed standards) and PAULA. A TEI P5 schema for the annotation of syntactic information is presented in § 3, and a discussion of relative merits of these standards is given in § 4.

2.1 TIGER-XML

TIGER-XML (Mengel and Lezius 2000) is a *de facto* standard for XML annotation of treebanks. It is well documented[1] and exemplified, it has been adopted in various projects, and it was the starting point for the SynAF proposed standard.

In TIGER-XML, each sentence is represented as a `<graph>` consisting of `<terminals>` and `<nonterminals>`. The `<terminals>` element is a list of `<t>`erminals, with orthographic, morphosyntactic and other information represented in attributes. Morphosyntactic attributes and their possible values may be defined in corpus `<head>`er.

Similarly, `<nonterminals>` is a list of `<nt>` syntactic nodes. Within each node, `<edge>`s link the node to its immediate constituents (`<t>`s or `<nt>`s). Additional secondary edges (`<secedge>` elements within `<nt>`) may be used to represent co-reference or other non-constituency information.

There is a treebank search engine serving TIGER-XML corpora, TIGERSearch (Lezius 2002, König *et al.* 2003), and converters from TIGER-XML to other formats, including the PAULA format used by ANNIS2 (`http://www.sfb632.uni-potsdam.de/d1/annis/`) and the Poliqarp (Janus and Przepiórkowski 2007) format.

[1] `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html`.

2.2 SynAF and Related Standards

Two proposed ISO standards immediately relevant for syntactic annotation are Syntactic Annotation Format (SynAF; ISO 24615) and Morphological Annotation Format (MAF; ISO 24611). According to `http://www.iso.org/` both were at the DIS stage of ISO standard development[2] at the time of writing this paper, although only the CD versions were available free of charge at `http://www.tc37sc4.org/`. It is these CD versions, ISO:24615 2009 and ISO:24611 2005 (see the bibliography), that we refer to here.

While TIGER-XML provides a specific schema, "SynAF is dealing with the description of a *metamodel* for syntactic annotation" (ISO:24615 2009, p. 6; emphasis ours), where syntactic annotation includes both constituency and dependency marking. The SynAF metamodel is described within a page and a half (ISO:24615 2009, pp. 12–14) as a straightforward generalisation of TIGER-XML.[3] The three main classes are: *T_Node* (for terminal nodes), *NT_Node* (for non-terminal nodes) and *Edge* (for dependency edges between nodes). Both kinds of nodes are defined over spans of text.

According to SynAF, syntactic annotation is applied to MAF-annotated input. MAF (ISO:24611 2005) is a much more specific and mature standard than SynAF, providing various examples of the encoding of morphosyntactic information. The two main elements it provides are `<token>` and `<wordForm>`. Tokens may identify spans in an external file (stand-off annotation), or may be marked in the original document (embedded annotation). Token attributes such as `@form` and `@transcription` may be used to abstract over the sequence of characters marked as a `<token>`. The `@join` attribute may specify whether there is whitespace to the left or right of the token. Two or more `<token>`s may overlap, e.g., in the case of abbreviations, where the final dot may belong to the abbreviation `<token>` and also constitute a separate `<token>`.

Word forms are defined over tokens. In the default case, one `<wordForm>` corresponds to one `<token>` and adds information about the lemma, the morphosyntactic analysis, etc. This linguistic information may be encoded as attributes of `<wordForm>`s or as feature structures (ISO:24610-1 2005) within `<wordForm>`s.[4] Word forms may also be empty (as for *pro* or PRO in Chomsky's

[2]There are six stages of development of any ISO standard: 1) initial proposal of a new work item, 2) preparation of a Working Draft (WD), 3) production and acceptance of the Committee Draft (CD), 4) production and acceptance of the Draft International Standard (DIS), to be distributed to ISO member bodies for commenting and voting, 5) approval of the Final Draft International Standard (FDIS), which has to pass the final vote, and 6) the publication of the International Standard (IS).

[3]Unfortunately, most of ISO:24615 2009 is devoted to an annex containing a preliminary list of syntactic data categories, even though, in our opinion, the contentious issue of linguistic data categories should be kept separate from the relatively straightforward matter of defining a metamodel or specific XML encoding of constituency and dependency relations. The other annex contains "testsuites", i.e., examples of MAF and SynAF annotation, which contain errors (they are not well-formed XML) and need "to be considerably revised" (p. 63). In summary, the practical usefulness of the version of SynAF referred to here is still rather limited.

[4]MAF contains also recommendations on tagset specification and on the handling of various

Principles and Parameters), or they may consist of a number of possibly discontinuous tokens. One <token> may also give rise to a number of <wordForm>s (as, possibly, in the case of German *am*, French *auquel*, Italian *damelo*, English *wanna* or Polish *nań*, if they are analysed as single <token>s). Moreover, <wordForm>s may contain other <wordForm>s, e.g., for the purpose of representing various multi-word units, so the domain of applicability of MAF overlaps somewhat with the domain of SynAF. Word forms may also refer to an external lexicon for their definitions.

Another proposed ISO standard that should also be mentioned here is Linguistic Annotation Framework (LAF; ISO:24612 2008), which defines a generic graph-based pivot format, called GrAF, designed to facilitate comparison and exchange of data in various annotation formats.

2.3 PAULA

PAULA (Ger. *Potsdamer AUstauschformat für Linguistische Annotation*; Dipper 2005), a LAF-inspired format developed within the SFB 632 project in Potsdam and Berlin, is an example of a family of general encoding standards for the annotation of multi-modal data.[5]

In the PAULA data model there are objects (“markables”), various types of relations between them, and features of objects. Markables may be simple spans of text (<mark>) or abstract <struct>ures bearing <rel>ations to other markables. For example, a syntactic constituent with 3 immediate daughters (one word and two syntactic constituents) may be represented as follows:[6]

```
<struct id="syn2"> <!-- PAULA -->
 <rel id="rel3" type="head"    xlink:href="tok.xml#t6"/>
 <rel id="rel4" type="nonhead" xlink:href="#syn20"/>
 <rel id="rel6" type="nonhead" xlink:href="#syn21"/>
</struct>
```

This representation closely corresponds to the following representation in TIGER-XML, though PAULA’s <rel> is a generalisation of TIGER-XML’s <edge> and may be used for the representation of various types of relations.

```
<nt id="nt2"> <!-- TIGER-XML -->
 <edge label="head"    idref="#t6"/>
 <edge label="nonhead" idref="#nt20"/>
 <edge label="nonhead" idref="#nt21"/>
</nt>
```

Additionally, markables are associated with feature values via a PAULA-specific encoding of feature structures.

kinds of ambiguities, including structural ambiguities encoded as finite state automata.

[5]See Dipper *et al.* 2006 for references to other such largely graph-based encodings.

[6]This is a modification of an example from Dipper 2005.

3 TEI P5

The Text Encoding Initiative "was established in 1987 to develop, maintain, and promulgate hardware- and software-independent methods for encoding humanities data in electronic form" (`http://www.tei-c.org/`). It is a *de facto*, constantly maintained XML standard for encoding and documenting textual data, with an active community, detailed guidelines (Burnard and Bauman 2008) and supporting tools. Its recommendations for the encoding of linguistic information are limited, but it includes the ISO FSR standard for representing feature structures, which can be used to encode various kinds of information (cf., e.g., Witt *et al.* 2009).

There are some TEI-encoded morphosyntactically annotated corpora, but the impact of the current P5 version of TEI Guidelines, released in November 2007, has been rather limited so far. Probably the main reason for this state of affairs is the richness and versatility of TEI. Ideas useful for linguistically annotated corpora are scattered over the 1350-odd pages of the Guidelines, and usually there is more than one way of representing any given annotation, so designing a coherent and constrained TEI-conformant schema for linguistic corpora is a daunting task.

One such schema, indirectly based on an earlier version of TEI Guidelines, is XCES (Ide *et al.* 2000), an XML-ised version of the TEI-based Corpus Encoding Standard (CES; Ide and Priest-Dorman 1995, Ide 1998) specified in SGML. XCES DTD schemata specify the representation of metadata, primary data, morphosyntactic annotation and — for parallel corpora — alignment. There are general feature structure mechanisms for the representation of other levels of information, but it is the specificity at the morphosyntactic and alignment levels that had a large influence on the relative success of that version of XCES. Around 2003 a new version of XCES was introduced — given as XML Schema specifications — that was a step back in this respect, as it lacks specific recommendations for any linguistic levels, resorting instead to general feature structure mechanisms; only minor technical modifications have been made to these schemata since their introduction. Other reasons why XCES does not currently meet the expectations of corpus developers are: 1) lack of documentation; `http://www.xces.org/` refers to old CES documentation as "supporting general encoding practices for linguistic corpora and tag usage" and "largely relevant to the XCES instantiation", although the CES documentation is hardly applicable to the second version of XCES, 2) feature structure mechanisms different from the established feature structure representation ISO standard (ISO:24610-1 2005), 3) lack of mechanisms for the encoding of discontinuity 4) or alternatives, and 5) the potential for confusion regarding the version of the standard (in particular, for many years DTD and XML Schema specifications co-existed on XCES web pages, without any clear information that they specify different representations).

In Przepiórkowski and Bański 2009b, we propose a representation of the primary data, text structure, text headers and corpus headers conformant with TEI P5. In this section we describe a possible TEI P5 encoding of syntactic information designed to maximise compatibility with other proposed standards.[7]

3.1 Morphosyntax and Other Assumptions

Following the common standard practice, we assume that each linguistic level is represented in its own file, referring to lower layers, down to the primary text, i.e., we assume the stand-off approach to annotation. More precisely, each corpus text is represented as a collection of files containing various annotation layers of the text, and each layer has the following minimal general structure (where `corpus_header.xml` contains the unique corpus header, and `header.xml` is the header of the particular text):

```
<teiCorpus
 xmlns:xi="http://www.w3.org/2001/XInclude"
 xmlns="http://www.tei-c.org/ns/1.0">
 <xi:include href="corpus_header.xml"/>
 <TEI>
  <xi:include href="header.xml"/>
  <text>
   <body><!-- text to annotate --></body>
  </text>
 </TEI>
</teiCorpus>
```

Depending on the type of text (written or spoken), `<body>` contains a list of `<p>`aragraphs (or possibly paragraph-length anonymous blocks, `<ab>`) or `<u>`tterance turns, further split into `<s>`entences, e.g.:

```
<body>
 <p xml:id="segm_p1">
  <s xml:id="segm_s1">...</s> <!-- more sentences here -->
 </p> <!-- more paragraphs here -->
</body>
```

We assume that this structure, down to the sentence level, is preserved and made parallel at all annotation layers. The correspondence between different layers is expressed with the TEI attribute `@corresp`. For example, assuming that the above code is a fragment of the segmentation layer of a text (by assumption represented in `ann_segmentation.xml`), defining word-level tokens, the next layer, morphosyntax, may have the following parallel structure:

```
<body>
 <p xml:id="morph_p1" corresp="ann_segmentation.xml#segm_p1">
  <s xml:id="morph_s1" corresp="ann_segmentation.xml#segm_s1">...</s>
 </p>
</body>
```

Whatever other annotation layers are present in the corpus, we assume the existence of a morphosyntactic layer (by assumption encoded in

[7]The overall picture, encompassing all linguistic levels assumed in the National Corpus of Polish, is presented in Przepiórkowski and Bański 2009a.

`ann_morphosyntax.xml`). Each sentence at this layer is a sequence of `<seg>` elements implicitly (via a specification in the schema) marked as `@type="token"`, and each `<seg>` contains a feature structure specification of various morphosyntactic information about the segment, e.g.:

```
<s xml:id="morph_s1" corresp="ann_segmentation.xml#segm_s1">
 <seg xml:id="morph_seg1"><fs>...</fs></seg>
 <seg xml:id="morph_seg2"><fs>...</fs></seg>
<!-- more segments here -->
</s>
```

TEI P5 contains wholesale the ISO standard for feature structure representation (ISO:24610-1 2005). In the interest of brevity and readability, we will not fully specify the XML encoding of feature structures, signalled above as `<fs>...</fs>`, but rather represent feature structures in a way common in linguistic theories such as HPSG and LFG. For example, the Polish segment *komputerem*, analysed as a singular instrumental inanimate-masculine form of the noun KOMPUTER, may have the following feature structure representation:

$$\begin{bmatrix} \textit{morph} & \\ \textsc{orth} & \text{komputerem} \\ \textsc{base} & \text{komputer} \\ \textsc{ctag} & \text{subst} \\ \textsc{msd} & \text{sg:inst:m3} \end{bmatrix}$$

Note that the names of features ORTH, BASE, CTAG and MSD are taken from (X)CES. Of course, other kinds of information may be represented here as well, including the whole list of possible interpretations (not just the one interpretation selected in the context), information about person or tool responsible for disambiguation, etc.

3.2 Representing Constituency

At the syntactic level (by assumption, in `ann_syntax.xml`), each `<s>`entence is a sequence of `<seg>` elements implicitly marked as `@type="group"`. More generally, for reasons of uniformity, we propose to use the `<seg>` element with different values of `@type` for different kinds of linguistic units (see § 3.4 below), but more specific TEI elements could be used here instead, e.g., `<w>` for words, `<phr>` for syntactic phrases and `<cl>` for clauses.

Just like word segments, syntactic groups also contain feature structure descriptions: in the simplest case, such a description may consist of a single attribute-value pair naming the node (e.g., $[\,\textsc{name}\ \text{NP}\,]$), but it could also be an LFG f-structure or a full-fledged HPSG feature structure. Apart from a feature structure specification of the node, such a syntactic group element may contain any number of `<ptr>` elements pointing at the immediate constituents of the group, e.g.:

```
<seg xml:id="group2">
 <fs>...</fs>
 <ptr xml:id="ptr3" type="head" target="ann_morphosyntax.xml#seg6"/>
 <ptr xml:id="ptr4" type="nonhead" target="#group20"/>
 <ptr xml:id="ptr6" type="nonhead" target="#group21"/>
</seg>
```

Note that immediate constituents may be words specified at a different layer or other syntactic groups of the same layer. Any `<ptr>` element may specify the type of the constituency relation, e.g., `head` or `nonhead`, and each has an `@xml:id`, so that the relation may be referred to in the feature structure description of the node.

Note also that this schema allows for discontinuous constituents, as `<ptr>` elements within one `<seg>` do not have to point at neighbouring constituents. This freedom, combined with the representations for dependencies outlined in the following subsection, makes it possible to encode various linguistic analyses of possible conflicts between phrase structure and dependency, e.g., involving extraposed material or crossing dependencies.

3.3 Representing Dependency

It is equally straightforward to represent dependency relations in TEI P5: instead of the one way `<ptr>` pointer used for immediate constituency, the `<link>` element may be used to relate two syntactic nodes (words or groups). According to TEI Guidelines, `<link>` may be used to represent symmetrical (bidirectional) or asymmetrical (unidirectional) relations; here, by convention, `<link>` represents asymmetrical edges in the dependency graph, whose end vertices are specified in the value of the attribute `@targets`.[8]

```
<seg xml:id="group43"><fs>...</fs></seg>
<link xml:id="link17" type="subject"
 targets="ann_morphosyntax.xml#seg78 #group43"/>
```

[8]The constraint that there be exactly two references within the values of `@targets` may be specified in RelaxNG by constraining the TEI data model (`http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-link.html`) from:

```
<rng:attribute name="targets">
 <rng:list>
  <rng:ref name="data.pointer"/>
  <rng:oneOrMore>
   <rng:ref name="data.pointer"/>
  </rng:oneOrMore>
 </rng:list>
</rng:attribute>
```

to:

```
<rng:attribute name="targets">
 <rng:list>
  <rng:ref name="data.pointer"/>
  <rng:ref name="data.pointer"/>
 </rng:list>
</rng:attribute>
```

Again, the value of `@type` specifies the kind of dependency. According to the example above, the dependency of type `subject` holds between a word (defined in `ann_morphosyntax`) and a syntactic group (defined elsewhere in the same file). In a “pure” dependency treebank, where dependencies are strictly between words, `<seg>` elements may be completely absent in this layer.

3.4 Case Study: National Corpus of Polish

It is not a prerequisite of the scheme proposed here that there be exactly two layers of grammatical representation of each text, `ann_morphosyntax.xml` and `ann_syntax`, and neither are fully rooted syntactic representations necessarily assumed here. Rather, there may be various layers of various granularity.

In the fully TEI P5-encoded National Corpus of Polish (Pol. *Narodowy Korpus Języka Polskiego*; NKJP; `http://nkjp.pl/`; Przepiórkowski *et al.* 2008, 2009), each text has the following linguistic layers: fine-grained word-level segmentation (including some segmentation ambiguities), morphosyntax (referring to disambiguated segmentation), coarse-grained syntactic words (e.g., for analytical tense forms consisting of multiple segments; referring to morphosyntax), named entities (referring to syntactic words) and syntactic groups (also referring to syntactic words).[9] The last layer assumes partial syntactic analysis, i.e., the annotation of nominal and other phrases, without the requirement that each word in the sentence must be contained in some syntactic group. All these layers are encoded as outlined in § 3.2, with different `@type`s of `<seg>` elements and different types of feature structure representations associated with `<seg>`s.

For example, at the syntactic words layer, `ann_words.xml`, each sentence consists of `<seg>` elements of `@type="word"` (vs. `@type="token"` for segmentation and morphosyntax). In the default case, a `<seg>` at this layer will be co-extensive with a `<seg>` at the lower (morphosyntax) layer, but it may also correspond to a possibly discontinuous list of morphosyntactic `<seg>`ments. Two different syntactic words may also overlap, as in *Bał się zaśmiać* ‘(He) feared (to) laugh’, where for two inherently reflexive verbs, BAĆ SIĘ ‘fear’ and ZAŚMIAĆ SIĘ ‘laugh’, one occurrence of the reflexive marker *się* suffices.[10] This situation is exemplified below:

```
<seg xml:id="word13">
 <fs>1</fs> <!-- (see below) -->
 <ptr target="ann_morphosyntax.xml#seg17"/> <!-- bał -->
 <ptr target="ann_morphosyntax.xml#seg18"/> <!-- się -->
</seg>
<seg xml:id="word14">
 <fs>2</fs> <!-- (see below) -->
 <ptr target="ann_morphosyntax.xml#seg18"/> <!-- się -->
 <ptr target="ann_morphosyntax.xml#seg19"/> <!-- zaśmiać -->
</seg>
```

[9]We ignore here another layer present in NKJP, that of word senses.

[10]On the haplology of the reflexive marker in Polish, see Kupść 1999.

$$\boxed{1} = \begin{bmatrix} \textit{word} & \\ \textsc{orth} & \text{bał się} \\ \textsc{base} & \text{bać się} \\ \textsc{ctag} & \text{Verbfin} \\ \textsc{msd} & \text{sg:ter:m1:imperf:past:ind:aff:refl} \end{bmatrix} \quad \boxed{2} = \begin{bmatrix} \textit{word} & \\ \textsc{orth} & \text{się zaśmiać} \\ \textsc{base} & \text{zaśmiać się} \\ \textsc{ctag} & \text{Inf} \\ \textsc{msd} & \text{perf:aff:refl} \end{bmatrix}$$

4 Discussion

The schema proposed above is not supposed to be novel; on the contrary, it has been designed to be as simple as possible and to be maximally compatible with other proposed standards for the encoding of grammatical information, but also with the aim of avoiding the potential problems of the other proposals.

Just like SynAF and PAULA, the schema is a straightforward extension of TIGER-XML: `<seg type="token">` (or `<seg type="word">`) elements directly correspond to TIGER's `<t>` and SynAF's *T_Node*, and `<seg type="group">` — to TIGER's `<nt>` and SynAF's *NT_Node*. Both kinds of `<seg>` elements correspond to PAULA's `<struct>`.

The schema maintains the distinction between two kinds of relations between syntactic nodes: immediate constituency, represented by `<ptr>`, and other — especially dependency — relations, represented by `<link>`, although in principle all kinds of relations could be represented via `<link>` elements, just as `<rel>` elements of different `@type`s represent different relations in PAULA. Compared to other standards, `<ptr>` corresponds to TIGER's `<edge>` and seems to have no analogue in SynAF, where apparently constituency is represented implicitly by the extent of the span of particular constituents.[11] On the other hand, `<link>` directly corresponds to SynAF's *Edge*[12] and is a generalisation of TIGER's `<secedge>`.

Moreover, various types of relations between `<token>`s and `<wordForm>`s, as defined in MAF, may be represented as illustrated in § 3.4. The current schema is also compatible with XCES, to the extent that XCES is originally TEI-based and given that the morphosyntactic representation outlined above uses XCES-inspired feature names.

We claim that the current schema inherits all advantages of various proposed standards, but improves on each of them. First of all, where TIGER-XML and MAF assume that different logical layers are present in the same file (words and syntactic groups in TIGER-XML, tokens and word forms in MAF), the schema proposed here assumes the stand-off philosophy of separating different layers of linguistic annotation.[13] Second, unlike TIGER-XML, which does not employ any feature structure representation, and unlike XCES and PAULA, which use non-standard feature structure representations, the schema proposed above complies

[11]The version of SynAF referred to in this paper is vague on this and various other specific issues.

[12]Although, curiously, in the Annex B of ISO:24615 2009, `<edge>` elements specify only one end of the edge.

[13]But such merging of annotation layers is still possible: `<seg>` elements of different `@type`s may occur in the same XML file.

fully with the ISO standard on feature structure representation. Moreover, unlike SynAF, whose current version seems to be an early draft, the schema is an application of TEI P5, a well-established and constantly maintained standard with stable guidelines and a large supporting community. In fact, since the schema is embedded in TEI, it is almost infinitely extendable and may draw from a variety of TEI solutions for various aspects of text representation.

5 Conclusion

One disadvantage of the Text Encoding Initiative P5 standard is that the documentation is huge and the task of distilling a manageable corpus schema is daunting. We have performed this task and reported the results in this and related papers (Przepiórkowski and Bański 2009a,b). Moreover, we looked at other proposed corpus encoding standards and concluded that whatever they offer may already be found in TEI, which has the advantage of being a very mature and at the same time actively supported standard. Nevertheless, whenever TEI provided alternative solutions, we chose mechanisms compatible with other proposed standards for treebank encoding, thus attaining a TEI schema maximally isomorphic with TIGER-XML, SynAF and PAULA. We hope that this work will serve as a starting point for the design of other TEI P5 corpus encoding schemata.

References

Bański, P. and Przepiórkowski, A. (2009). Stand-off TEI annotation: the case of the National Corpus of Polish. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III) at ACL-IJCNLP 2009*, pages 64–67, Singapore.

Burnard, L. and Bauman, S., editors (2008). *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Oxford. `http://www.tei-c.org/Guidelines/P5/`.

Dipper, S. (2005). Stand-off representation and exploitation of multi-level linguistic annotation. In *Proceedings of Berliner XML Tage 2005 (BXML 2005)*, pages 39–50, Berlin.

Dipper, S., Hinrichs, E., Schmidt, T., Wagner, A., and Witt, A. (2006). Sustainability of linguistic resources. In E. Hinrichs, N. Ide, M. Palmer, and J. Pustejovsky, editors, *Proceedings of the LREC 2006 Workshop on Merging and Layering Linguistic Information*, pages 14–18, Genoa. ELRA.

Ide, N. (1998). Corpus Encoding Standard: SGML guidelines for encoding linguistic corpora. In *Proceedings of the First International Conference on Language Resources and Evaluation, LREC 1998*, pages 463–470, Granada. ELRA.

Ide, N. and Priest-Dorman, G. (1995). Corpus encoding standard. `http://www.cs.vassar.edu/CES/`, accessed on 2009-08-22.

Ide, N., Bonhomme, P., and Romary, L. (2000). XCES: An XML-based standard for linguistic corpora. In LREC (2000), pages 825–830.

ISO:24610-1 (2005). Language resource management – feature structures – part 1: Feature structure representation. ISO/DIS 24610-1, 2005-10-20.

ISO:24611 (2005). Language resource management – Morpho-syntactic annotation framework (MAF). ISO/CD 24611, ISO TC 37/SC 4 document N 225 of 2005-10-15.

ISO:24612 (2008). Language resource management – Linguistic annotation framework. ISO/WD 2461[2], ISO TC 37/SC 4 document N 463 rev00 of 2008-05-12.

ISO:24615 (2009). Language resource management – Syntactic annotation framework (SynAF). ISO/CD 24615, ISO TC 37/SC 4 document N 421 of 2009-01-30.

Janus, D. and Przepiórkowski, A. (2007). Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 85–88, Prague.

Kupść, A. (1999). Haplology of the Polish reflexive marker. In R. D. Borsley and A. Przepiórkowski, editors, *Slavic in Head-Driven Phrase Structure Grammar*, pages 91–124. CSLI Publications, Stanford, CA.

König, E., Lezius, W., and Voormann, H. (2003). *TIGERSearch 2.1: User's Manual*. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Lezius, W. (2002). TIGERSearch — ein Suchwerkzeug für Baumbanken. In S. Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*, Saarbrücken.

LREC (2000). *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000*, Athens. ELRA.

Mengel, A. and Lezius, W. (2000). An XML-based encoding format for syntactically annotated corpora. In LREC (2000), pages 121–126.

Przepiórkowski, A. and Bański, P. (2009a). Which XML standards for multilevel corpus annotation? In *Proceedings of the 4th Language & Technology Conference*, Poznań, Poland. Forthcoming.

Przepiórkowski, A. and Bański, P. (2009b). XML text interchange format in the National Corpus of Polish. In S. Goźdź-Roszkowski, editor, *The proceedings of Practical Applications in Language and Computers PALC 2009*, Frankfurt am Main. Peter Lang. Forthcoming.

Przepiórkowski, A., Górski, R. L., Lewandowska-Tomaszczyk, B., and Łaziński, M. (2008). Towards the National Corpus of Polish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2008*, Marrakech. ELRA.

Przepiórkowski, A., Górski, R. L., Łaziński, M., and Pęzik, P. (2009). Recent developments in the National Corpus of Polish. In J. Levická and R. Garabík, editors, *Proceedings of Slovko 2009: Fifth International Conference on NLP, Corpus Linguistics, Corpus Based Grammar Research, 25–27 November 2009, Smolenice/Bratislava, Slovakia*, Brno. Tribun.

Witt, A., Rehm, G., Hinrichs, E., Lehmberg, T., and Stemann, J. (2009). SusTEInability of linguistic resources through feature structures. *Literary and Linguistic Computing*, **24**(3), 363–372.

MaJo - A Toolkit for Supervised Word Sense Disambiguation and Active Learning

Ines Rehbein   Josef Ruppenhofer   Jonas Sunde

Universität des Saarlandes, Germany
Computational Linguistics

{rehbein,josefr}@coli.uni-sb.de   s9sujona@stud.uni-saarland.de

Abstract

We present MaJo, a toolkit for supervised Word Sense Disambiguation (WSD), with an interface for Active Learning. Our toolkit combines a flexible plugin architecture which can easily be extended, with a graphical user interface which guides the user through the learning process. MaJo integrates off-the-shelf NLP tools like POS taggers, treebank-trained statistical parsers, as well as linguistic resources like WordNet and GermaNet. It enables the user to systematically explore the benefit gained from different feature types for WSD. In addition, MaJo provides an Active Learning environment, where the system presents carefully selected instances to a human oracle. The toolkit supports manual annotation of the selected instances and re-trains the system on the extended data set. MaJo also provides the means to evaluate the performance of the system against a gold standard.

We illustrate the usefulness of our system by learning the frames (word senses) for three verbs from the SALSA corpus, a version of the TiGer treebank with an additional layer of frame-semantic annotation. We show how MaJo can be used to tune the feature set for specific target words and so improve performance for these targets. We also show that syntactic features, when carefully tuned to the target word, can lead to a substantial increase in performance.

1 Introduction

An important step in Natural Language Processing is the disambiguation of word senses, without which we would not be able to interpret the meaning of an utterance or text. Word Sense Disambiguation (WSD) thus provides important information for many NLP applications in the area of information retrieval, summarisation, question answering or machine translation. To date, the best performance for the task of WSD is achieved by supervised systems which rely on manually labelled training data. However, there are two major drawbacks to the supervised approach:

1. Supervised learning requires a large amount of manually labelled data.

2. Supervised learning is highly sensitive to the domain the labelled data is taken from.

Manual annotation of large data sets is time-consuming and costly. Therefore it is infeasible to create resources a) which are large enough to capture all information needed for disambiguation and b) which are representative of all possible genres and domains we might want to work with. Active Learning is one possible approach to address this bottleneck.

Active Learning tries to reduce the amount of human annotation by carefully selecting new training instances with respect to the information content they provide for the machine learning classifier used in the supervised setting. These instances are then handed over to a human annotator, the oracle, who assigns the correct label. The basic idea is to select a small number of instances which provide important information for the classifier and to thereby achieve an increase in performance in the same range as would be achieved on a larger training set of randomly selected training examples.

Although Active Learning has been shown to be useful for WSD in general [7, 24], and in particular for tackling problems of domain adaptation [5], some open issues need to be addressed. One of them is the impact of feature design on the WSD task. Chen and Palmer [6] show that a set of rich linguistic features does improve the performance of WSD systems. Xue et al. [23] argue that word senses are partitioned along different dimensions for different verbs and that, as a consequence, we need to tune the set of features used for disambiguation for each particular target verb. What we do not know is which types of features are beneficial for which (types of) target verbs. A systematic investigation of different feature types such as syntactic and semantic features, context features, collocational features, and so on, is urgently needed.

Another issue concerns the Active Learning environment. Recent work has explored the impact of different parameters on the performance of Active Learning. Among these are the size of the seed data for initial training; different techniques for selecting new, informative examples to be labeled by the human oracle; sampling techniques for providing the system with new training instances to choose from; as well as developing stopping criteria for determining the optimal point to end the Active Learning process [24, 16, 22, 25, 2, 19]. Other issues which are also crucial for the Active Learning setup are the grain size of sense distinctions used for annotation as well as the distribution of the different word senses in the data. It is not yet clear whether Active Learning does work for coarse-grained word sense distinctions only [8], or whether it can also improve performance for fine-grained, detailed word senses [5].

In this paper, we provide a means for tackling these questions. We present MaJo, a toolkit with a graphical user interface for applying Active Learning to a Word Sense Disambiguation task. Our toolkit allows users to combine different components for syntactic and semantic pre-processing, to choose between different sets of features which can easily be adapted to the learning problem, and to integrate Active Learning in the training process. Our main intention in building Majo is to provide an explorative tool for investigating the contribution of individual features to the learning problem and to provide an easily accessible way to test the impact of Active Learning on different target words.

The remaining part of the paper is structured as follows. First we describe the architecture of our tool and show how it can be used for supervised WSD with and without Active Learning. Next we present experiments assessing the impact of syntactic and semantic features on the performace of our WSD system and show how MaJo can be used for tuning the feature set to particular target words. Finally we conclude and outline future work.

2 A Tool for Feature Exploration and Active Learning

The tool presented in the paper, MaJo, allows the user to explore the usefulness of different feature types for WSD in an Active Learning environment. The graphical user interface guides the user through the learning process and provides an easy way to include or exclude individual features for training. The ability to display the extracted features for all instances allows for a qualitative evaluation of the benefit obtained by individual features. In the Active Learning environment the user is presented with selected instances, which can comfortably be labelled in the GUI. The manually labelled instances are then added to the seed data, and the system is trained on the new data set.

2.1 Architecture

MaJo features a flexible plugin architecture which implements a number of interfaces to off-the-shelf NLP tools and linguistic resources for extracting training data from the web (Yahoo! search API), for preprocessing (Stanford POS Tagger [21], Stanford Parser [13], Berkeley Parser [20], MaltParser [18]), for extracting semantic features (WordNet [12], GermaNet [15]) and for classification (OpenNLP MaxEnt 2.5[1]). The architecture can easily be extended to incorporate additional components for preprocessing and feature extraction, or to implement new machine learning algorithms for training. At the moment the system provides working interfaces for English and German, but it can easily be extended to other languages.

2.2 Supervised Learning with MaJo

The GUI was designed to guide users with only basic computer skills through the training process. First, the user has to enter a target lemma he or she wants to train the system on. The system generates a list of possible inflected word forms for the target lemma, using a precompiled dictionary[2]. The user can remove unwanted

[1]`http://maxent.sourceforge.net`

[2]The German dictionary was created with Morphy, a morphological analyser by Wolfgang Lezius [17]

word forms or add new ones to the dictionary. Next, the user is asked to load a text file with annotated training data or to enter new, labelled instances for training in a text field. After that, the user can choose the plugins for preprocessing and feature selection.

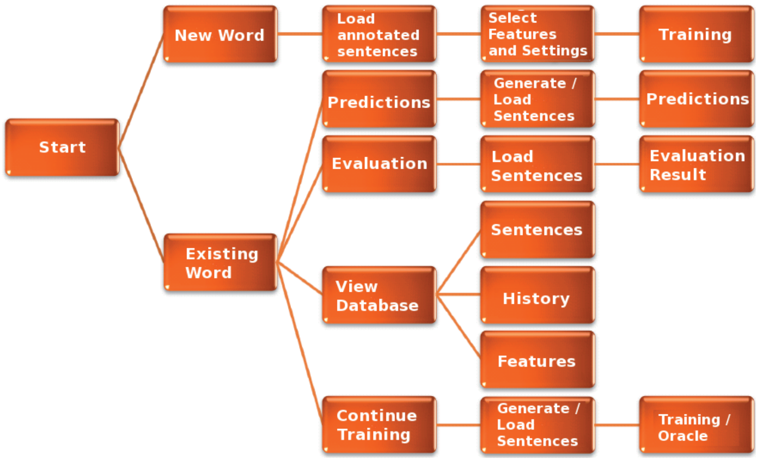


Figure 1: Working stages during supervised learning with and without Active Learning

Plugins

At the moment, our system provides the plugins listed in Table 1. Feature classes (1) and (2) are bag-of-word context features extracting the word form (1) or POS tag (2) for each word occuring in a context window of size *n*. Feature class (3) relies on information about syntactic categories provided by either the Berkeley Parser or the Stanford Parser, which have been trained on a constituent version of the TiGer treebank [3]. The user can specify a syntactic category, which is considered as context for which all child nodes (word form or POS tags) are extracted. Feature class (4) is based on functional dependency information provided by the MaltParser, which was trained on a dependency version of TiGer. Feature classes (5) and (6) add semantic information to the feature set, based on different preprocessing steps. For (5), we extract selected semantic relations like hyperonymy or meronymy for specified POS tags. The WordNet(/GermaNet)POSTag plugin uses a more fine-grained POS tag annotation, while the WordNet(/GermaNet)SuperordinateTag plugin uses a more coarse-grained (super-ordinate) POS tag scheme. For (6), semantic relations are extracted for specific functional dependencies as provided by the MaltParser.

After having selected the feature set and a machine learning classifier for training[3], the system starts feature extraction and training on the annotated training

[3]At present, the system implements the OpenNLP MaxEnt classifier. We plan to integrate other ML algorithms in the future.

	Feature Class	Description	Parameter	
(1)	WordRangeContext	bag-of-word context	window size	
(2)	POSTagContext	bag-of-POS-tag context	window size	Berk./Stan. POS Tagger
(3)	ClauseFunDep	words or POS tags for given functional dependencies	functional dependency	MaltParser
(4)	SentencePhrase FunDep	words or POS tags for children of a specific syntactic category	syntactic category	Berk./Stan. Parser
(5)	WordNet/GermaNet (Super)POSTag	WordNet relations for (super-ordinate) POS tags	max. depth, sem. relation	Berk./Stan. POS Tagger
(6)	WordNet/GermaNet FunDep	WordNet relations for specific functional dependencies	max. depth, sem. relation	MaltParser

Table 1: Off-the-shelf software components implemented in MaJo

data. Having finished the training process, the training data is stored in a database for future use. Now the user can access the database and evaluate the performance of the system against a gold standard (see Figure 2.2).

Accessing the database also offers other options to the user. It is possible to predict word senses for new, unannotated text which can be a) loaded from a text file, b) entered in a text field (GUI) or c) generated using the Yahoo! search interface. Another option allows the user to display the data stored in the database. For each sentence in the database the user can check which features have been extracted by the different plugins. This provides the means for a qualitative evaluation of the usefulness of individual features.

2.3 Active Learning

The last option offered by the system provides the environment for Active Learning. The pre-stored data in the database can be considered as seed data, which is used by the ML classifier to predict labels for new, unannotated text. These newly annotated sentences can serve as a pool for selecting instances for Active Learning.

As described above, for Active Learning we need to identify those instances which are the most informative for the classifier. The user can define a threshold for selecting new instances, based on the confidence score of the classifier. The confidence score reported by the maximum entropy classifier specifies the probability that instance n is assigned label x. We can use this score to determine which instances the classifier is most uncertain of. The intution behind this is that the classifier has yet to learn how to label these instances. Therefore we select new training instances by setting a confidence threshold, so that all instances below the threshold will be presented to the oracle to be manually disambiguated, and then added to the training set. The GUI provides a comfortable interface for the human annotator, who can then chose the correct label for each of the selected sentences from a pulldown menu. When the annotation is finished, the system is retrained on the new data set, consisting of the seed data and the newly added, manually labelled

freq	**bringen**	freq	**gewinnen**	freq	**drohen**
2	Position_on_a_scale				
3	Erbringen				
5	Achieve_status				
6	Deprive				
6	Put_behind				
9	Accumulated_amount				
12	Contribute_effort	3	Manufacturing		
15	Formulation	4	Improvement		
21	Entail	30	Bring_about_result		
39	Befall_patient	37	Change_position_on_a_scale		
40	Present	40	Win_prize		
105	Bringing	43	Come_to_be_in_state	243	drohen1-salsa
112	Receive_caused_experience	43	Suasion	256	Commitment
471	Cause_patient_to_be_in_state	300	Win_competition	501	Run_risk
850	train/test: 680/170	500	train/test: 400/100	1000	train/test: 800/200

Table 2: Word senses for *bringen* (to bring), *gewinnen* (to win) and *drohen* (to threaten)

instances. After retraining, the user has the option to evaluate the performance of the new training set or to continue the Active Learning process.

3 Experiments

In our experiments we want to assess the impact of different types of features on a WSD task for German verbs. We chose the three German verbs *drohen* (threaten), *gewinnen* (win) and *bringen* (bring), because they are reasonably frequent and exhibit a range of difficulty in terms of the number of word senses. Our sense inventory follows the SALSA annotation scheme [4]. The SALSA corpus is a frame-semantic lexical resource for German, adding an additional layer of semantic annotation to the TiGer treebank. Semantic frames can be considered as word senses, and so the task of frame assignment is basically a WSD problem [10].

Table 2 shows the number of instances for the three verbs in our data set as well as the different word senses and their distribution. We performed a 5-fold cross-validation, randomly generating training and test folds from the pool of available instances. The main objective of our experiments is to investigate the impact of different feature types on the WSD task for the three target verbs. We want to test how much we can gain in terms of precision and recall by tuning the feature set to the individual target verbs. Furthermore, we want to test which types of features are helpful for the different targets. What can we expect when applying shallow context features only, and how much can be gained by adding syntactic and semantic features to the feature set?

3.1 Results

In our experiments we first tested the performance of our system when trained with individual features. We report f-scores averaged over the 5 folds for each

Feature	**drohen**	**gewinnen**	**bringen**
A	*context features (word form)*		
wordRange 2	0.689	0.588	**0.570**
wordRange 5	**0.702**	**0.640**	0.550
wordRange 8	0.684	0.632	0.540
B	*context features (POS tags)*		
StanPOS 2	0.617	0.540	**0.500**
StanPOS 3	**0.651**	0.490	0.490
StanPOS 3, +PUNC	0.634	**0.550**	0.470
C	*Word form/POS tag context for specific syntactic categories*		
SentPhrase NP	0.491	0.550	0.430
SentPhrase VP	**0.617**	**0.642**	**0.530**
SentPhrasePOS NP	0.494	**0.598**	**0.500**
SentPhrasePOS VP	**0.621**	0.594	0.490
D	*GermaNet semantic relations for superordinate POS tags, depth 3*		
hyper, meron, N	0.570	0.612	0.530
hyper, meron, V	0.582	0.566	0.510
hyper, meron, NAV	**0.593**	**0.626**	**0.550**
E	*GermaNet semantic relations for GF (MaltParser), depth 3*		
SUBJ, OBJA, OBJD, hyper	0.499	0.612	0.510
SUBJ, OBJA, OBJD, OBJG hyper	0.510	**0.614**	0.520
SUBJ, OBJA, OBJD, hyper, meron	**0.528**	0.610	**0.550**
F	*Combinations of the best features for each target verb*		
best settings	0.701	0.650	0.560

Table 3: Results for individual feature plugins and combinations thereof

target word.

Context features

First, we trained the system using shallow context features. We extracted the word forms in a context window of size 2, 5 and 8 to the left and right of the target word. Table 3 *A* shows results for our three targets. For *drohen* and *gewinnen* we obtain best results with a context window of size 5, while for *bringen* a smaller window (size 2) shows better performance.
When extracting POS tag context features, for *drohen* best results are achieved with a context window of size 3. For *gewinnen*, including punctuation in the feature set brings a considerable improvement of 6%, while for *drohen* and *bringen* results decrease (Table 3, *B*). *Bringen*, however, achieves best results with a small window size of 2, as was the case for the word range context. Using word forms as features by far outperforms POS tag context features for all three verbs.

Syntactic features

Using syntactic categories to specify the context window for which we extract word forms, we obtain by far better results when selecting verb phrases as relevant context. For POS tag features, the verb *drohen* again benefits from the VP context (with an improved performance of around 12% over NP context), while for *gewinnen* and *bringen* we observe a slightly better performance when using NPs as context (Table 3, *C*).

Semantic features

Using semantic features as the only clue for WSD, again we obtain mixed results. Table 3, D shows results for extracting semantic relations from Germanet for superordinate POS tags (nouns N, verbs V, adjectives ADJ). While for *drohen* the contribution of the GermaNet features is less than that of the shallow context features or the syntactically motivated context features, for *gewinnen* and *bringen* the semantic features contain more relevant clues for the disambiguation process. However, they are still outperformed by the word context features which, for all three targets, obtain best results.

Our last feature class, E, extracts semantic relations from GermaNet for specific grammatical functions assigned by the MaltParser. Surprisingly, selecting features for functional dependents like subject, accusative object and dative object does not yield better results than extracting the same relations for all nouns, verbs and adjectives in the sentence. One possible explanation is the low performance for grammatical function labelling in German parsing. For subjects, results are quite high with 90.2% f-score, for accusative objects, however, we see only a performance of 80.0%, and for dative objects the MaltParser achieves 49.7% f-score only [14]. As a result, the classifier has to deal with a great amount of noise in the feature set, which might be responsible for the poor results for feature class E.

Combined features

Next, we selected the parameter setting for which we obtained best performance for each feature class, and trained the system on the combined feature set (Table 3, F) Surprisingly, performance for the combined features is not much higher than for the best individual feature classes. We suspect that by selecting the best performing feature setting we select features capturing the same kind of information, so that the benefit obtained from the additional features is not very great. By using a large feature set, on the other hand, we also add more noise to the data.

Dinu and Kübler [9] argue that, at least for memory-based learning, a reduced and controlled amount of features is more beneficial than the the full range of features that have been proposed in the WSD literature so far. We follow this notion and restrict our feature set to a subset of linguistically motivated features, based on an analysis of the specific sense distinctions for our target verbs. The *History* option provided by MaJo allows the user to inspect the features extracted by each of the plugins, and to use the information for feature tuning. For *drohen*, the word sense distinctions are more syntactically motivated than for *gewinnen* or *bringen*, which becomes apparent when looking at the results for the individual feature classes. The contribution of the semantic feature classes for *drohen* is rather small (as compared to results for *gewinnen* and *bringen*), while overall results for *drohen* are higher than for the other two targets.

The new set of features resulting from our analysis[4] achieves an f-score of 0.775, which is significantly higher than the result obtained by combining the sin-

[4]ClauseFunDep (Table 1, ROOT, SUBJ, OBJA, OBJD, PP, OBJP, AUX, PART, AVZ, ATTR), PosTagContext (window size 3, no punctuation, Stanford POS Tagger), WordRangeContext (window size 5), SentencePhrasePOS (Berkeley Parser, VZ)

gle best-performing feature-classes. This shows that syntactic features (as well as the semantic features based on the output of a treebank-trained dependency parser) can yield a substantial improvement when choosing the right settings for feature selection as well as appropriate features.

To compare our tool with a state-of-the-art WSD system, we trained MaJo on all instances in our *drohen* data set and tested it on a test set with 111 instances taken from the SALSA corpus. We also trained Shalmaneser [11], a shallow semantic parser, on the same training set and run it on our test set. MaJo achieves an f-score of 0.712 on the SALSA test set, while the performance of Shalmaneser is much lower with 0.362. Please note that these results are not representative for the overall performance of the two systems. For a fair comparison we need to test both systems on a larger number of target words, which is beyond the scope of this paper.

4 Discussion

The results that we have obtained match well with our linguistic intuitions about the relative difficulty of our three verbs. *Bringen* as the most polysemous verb is the most difficult, followed by *gewinnen*, and *drohen*. However, the number of senses by itself is not the whole story. Also relevant are the actual distribution of senses in the data and the ability to extract useful features.

Consider, as an example, the senses of the verb *bringen*. Several of the senses, the ones shown in Table (4), are actually multi-word expressions with easily identifiable components such as the expletive object *es*, PPs headed by specific prepositions or with reflexive pronouns as prepositional objects. However, these senses make up a mere 9% of our data. The by far most common sense, Cause_patient-_to_be_in_state, expresses a very general meaning of causation and typically occurs in the pattern *NP_acc PP bringen*, with a lot of different prepositions heading the secondary predicate expressing the caused state of affairs. The two next most frequent senses, the basic sense of Bringing and another metaphorical causation sense, Receive_caused_experience, share the same basic syntactic configuration *NP_dat NP_acc bringen* and are most readily distinguished by the semantics of the accusative object. Overall, the semantic deck is stacked against us: the frequent senses are syntactically not very distinctive, whereas the distinctive senses are not very frequent. In this regard, *bringen* contrasts rather strongly with our best-performing lemma, *drohen*. Its three senses have distinct prototypical syntactic realizations. Commitment clauses have the two forms *NP_nom drohen NP_dat PP_mit(with)* and *NP_nom drohen NP_dat VP*; drohen1-salsa has the form *NP_dat drohen NP_nom*; and Run_risk typically has the form *NP_nom drohen VP*. Importantly, the skew in the frequency of these senses is not very great, with one sense accounting for half of the tokens and the others for a quarter each.

Frame	Canonical Form	Freq.
Achieve_status	es PP_zu etwas bringen	5
Deprive	NP_acc PP_um bringen	6
Put_behind	NP_acc hinter pron_refl. bringen	21
Accumulated_amount	es PP_auf bringen	9
Formulation	NP_acc auf einen Nenner bringen	15
Entail	NP_ mit sich bringen	21
Total		77

Table 4: Easily identifiable senses of *bringen*

5 Conclusions and Future Work

We presented MaJo, a toolkit for supervised WSD, which incorporates an environment for Active Learning.[5] Our tool provides an easy-to-use GUI and is tailored to support feature tuning for specific target words such as we carried out here. Our experiments showed that, even for medium-sized data sets, much can be gained by tuning the feature set to specific target words, and that especially the syntactic and semantic feature types can bring a significant improvement, provided that we use the right features.

In future work we will extend the feature classes used for disambiguation as well as the options for the Active Learning environment, and integrate additional ML classifiers. We also plan to use the tool to study the interaction between the criteria on which sense distinctions are based and the learnability for automatic systems of these distinctions, comparing for instance FrameNet[1] sense distinctions with those of WordNet.

References

[1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of the COLING-ACL*, pages 86–90, 1998.

[2] Michael Bloodgood and Vijay Shanker. A method for stopping Active Learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, Boulder, Colorado, 2009.

[3] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In Erhard W. Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 24–42, Sozopol, Bulgaria, 2002.

[5]MaJo is freely available for research purposes and can be downloaded from `http://www.coli.uni-saarland.de/projects/salsa`.

[4] A. Burchardt, K. Erk, A. Frank, A. Kowalski, S. Padó, and M. Pinkal. The SALSA Corpus: a German corpus resource for Lexical Semantics. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, Genoa, Italy, 2006.

[5] Yee Seng Chan and Hwee Tou Ng. Domain adaptation with Active Learning for Word Sense Disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[6] Jinying Chen and Martha Stone Palmer. Towards robust high performance Word Sense Disambiguation of English verbs using rich linguistic features. In *IJCNLP*, 2005.

[7] Jinying Chen, Andrew Schein, Lyle Ungar, and Martha Palmer. An empirical study of the behavior of Active Learning for Word Sense Disambiguation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, New York, NY, 2006.

[8] Hoa Trang Dang. *Investigations into the role of Lexical Semantics in Word Sense Disambiguation*. PhD dissertation, University of Pennsylvania, Pennsylvania, PA, 2004.

[9] Georgiana Dinu and Sandra Kübler. Sometimes less is more: Romanian Word Sense Disambiguation revisited. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-07)*, Borovets, Bulgaria, 2007.

[10] Katrin Erk. Frame assignment as Word Sense Disambiguation. In *Proceedings of the 6th International Workshop on Computational Semantics (IWCS-6)*, Tilburg, The Netherlands, 2005.

[11] Katrin Erk and Sebastian Pado. Shalmaneser - a flexible toolbox for Semantic Role assignment. In *Proceedings of LREC*, Genoa, Italy, 2006.

[12] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press, first edition, 1998.

[13] Dan Klein and Chris Manning. Accurate unlexicalized parsing. In *41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, Sapporo, Japan, 2003.

[14] Sandra Kübler. The PaGe 2008 shared task on parsing German. In *ACL Workshop on Parsing German (PaGe-08)*, Columbus, OH, 2008.

[15] Claudia Kunze and Lothar Lemnitzer. GermaNet - representation, visualization, application. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, 2002.

[16] Florian Laws and Hinrich Schütze. Stopping criteria for Active Learning of Named Entity Recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK, August 2008.

[17] Wolfgang Lezius. Morphy - German morphology, part-of-speech tagging and applications. In *Proceedings of the 9th EURALEX International Congress*, Stuttgart, Germany, 2000.

[18] Joakim Nivre, Johan Hall, and Jens Nilsson. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, Genoa, Italy, 2006.

[19] Fredrik Olsson and Katrin Tomanek. An intrinsic stopping criterion for committee-based Active Learning. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, Boulder, Colorado, 2009.

[20] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference and the 7 th Annual Meeting of the North American Chapter of the Association for Computati onal Linguistics (HLT-NAACL-07)*, Rochester, NY, 2007.

[21] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-03)*, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[22] Andreas Vlachos. A stopping criterion for Active Learning. *Compututer Speech and Language*, 22(3), 2008.

[23] Nianwen Xue, Jinying Chen, and Martha Palmer. Aligning features with sense distinction dimensions. In *ACL*, 2006.

[24] Jingbo Zhu and Eduard Hovy. Active Learning for Word Sense Disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, 2007.

[25] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active Learning with sampling by uncertainty and density for Word Sense Disambiguation and Text Classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK, 2008.

An English Dependency Treebank à la Tesnière

Federico Sangati
University of Amsterdam
f.sangati@uva.nl

Chiara Mazza
University of Pisa
chiara.mazza@gmail.com

Abstract

During the last decade, the Computational Linguistics community has shown an increased interest in Dependency Treebanks. Several groups have developed new annotated corpora using dependency representation, while other people have proposed several automatic conversion algorithms to transform available Phrase Structure (PS) treebanks into Dependency Structure (DS) notation. Such projects typically refer to Tesnière as the father of dependency syntax, but little attempt has been made to explain how the chosen representation relates to the original work. A careful comparison reveals substantial differences: modern DS annotations discard some relevant features characterizing Tesnière's model. This paper is presenting our attempt to go back to the roots of dependency theory, and show how it is possible to transform a PS English treebank to a DS notation that is closer to the one proposed by Tesnière, which we will refer to as TDS. We will show how this representation can incorporate all main advantages of modern DS, while avoiding well known problems concerning the choice of heads, and better representing common linguistic phenomena such as coordination.

1 Introduction

Although the tradition of using syntactic models in linguistics can be dated back to Panini's work (4th century BC), the discussion about which model linguists should use is still open. Corpus-based Computational Linguistics (CL) was born in the middle of last century, an important period of change in linguistic theory. In continental Europe, the French structuralist Lucien Tesnière, was developing a general theory of syntax, which was published posthumously in 1959 [15], and would became the foundation of Dependency Structure (DS) theory. In 1957 Noam Chomsky published his own work on Syntactic Structures [3], which would become the main reference for Phrase Structure (PS) theory.

Corpus-based CL has been strongly influenced by PS notation, due to the strong worldwide position of Chomskyan theory, and still nowadays most linguistic resources conform to this representation. Only in the last decade, more and more interest was placed on dependency theory, thanks to several research groups that have developed new annotated treebanks using DS formalisms (e.g., [1], [6]). In the same period other people have proposed several automatic conversion algorithms to transform available PS treebanks into DS (e.g., [17], [5], [9]). Such projects typically refer to Tesnière as the father of dependency syntax, but little attempt has been made to explain how the chosen dependency representation relates to the original work. A careful comparison reveals substantial differences: modern DS retains only the main idea proposed by Tesnière, namely the relation of dependency between words (section 2.1), while other operations and features of the original model are discarded or not overtly represented.

In the current paper we present an ongoing project of converting the English Penn Wall Street Journal (WSJ) Treebank [11] into a DS notation that is closer to the one proposed by Tesnière, which we will refer to as TDS. In particular we will reintroduce three key concepts: the division of a sentence into *blocks* of words, which act as intermediate linguistic units (section 2.2), the *junction* operation, to handle coordination and other types of conjoined structures (section 2.3), and the operation of *transference*, to generalize over the categories of the linguistic elements (section 2.4)[1].

Our work is not purely driven by historical concerns; in section 3 we will in fact give empirical evidence that shows how this representation can incorporate all main advantages of modern DS, while avoiding well known problems concerning the choice of heads, and better representing common linguistic phenomena such as coordination. Finally, in section 4 we will give more details about the conversion procedure, and the generated structures.

2 Dependency Structures à la Tesnière

2.1 The dependency relation

The main idea behind Tesnière's model is the notion of *dependency*, which identifies the syntactic relation existing between two elements within a sentence, one of them taking the role of governor (or head) and the other of dependent (*régissant* and *subordonné* in the original terminology). Tesnière schematizes this syntactic relation using a TDS *stemma* as in figure 1, putting the governors above the dependents. On the right side of the figure we present the same sentence using our notation, incorporating all the main features introduced by Tesnière, which we will explain in the following sections.

[1]See in [15] part I ch. 22 on *nucléus*, part II on *jonction*, and part III on *translation*. We choose *transference* for the original French word *translation* to avoid any misunderstanding with the other meaning of the word *translation* in English. Unfortunately, 50 years after its publication, there is still no official English translation of the author's work.

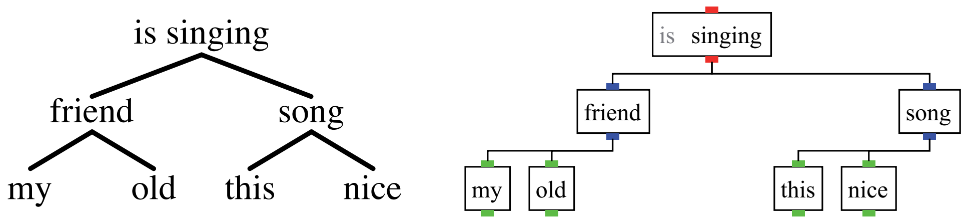


Figure 1: TDS of the sentence "*My old friend is singing this nice song*", in Tesnière notation (left) and in our TDS representation (right).

2.2 Words, blocks and categories

In TDS, all words are divided into two classes: full content words (e.g., nouns, verbs, adjectives, etc.), and empty functional words (e.g., determiners, prepositions, etc.). Each full word forms a *block*[2] which may additionally include one or more empty words, and it is on blocks that operations are applied. Tesnière distinguishes four *block categories* (or functional labels[3]), here listed together with the original single letter notation, and the color reported in our diagrams: *nouns* (O, blue), *adjectives* (A, green), *verbs* (I, red), and *adverbs* (E, yellow).

The verb represents the process expressed by the clause, and all its arguments, representing the participants in the process, have the functional labels of nouns, and are determined by the *valence* of the verb. On the other hand the verb's adjuncts (or circonstants), representing the circumstances under which the process is taking place, i.e., time, manner, location, etc., have the functional labels of adverbs. We will now introduces two operations, *junction* and *transference*, by means of which it is possible to construct more complex clauses from simple ones.

2.3 Junction

The first operation is the *junction*. It is employed to group blocks which lie at the same level, the *conjuncts*, into a unique entity which has the status of a block. The conjuncts are horizontally connected in the TDS, belong to the same category, and are possibly (and not always) connected by means of empty words, the *conjunctions*. Figure 2 displays three coordinated structures[4].

[2]Tesnière in [15, ch. 22] uses the term *nucléus*, and explains that it is important in order to define, among other things, the *transference* operation (see section 2.4). In our diagrams blocks are represented as black boxes, and empty words are written in grey to distinguish them from full words.

[3]We will use both terms interchangeably. Categories can be roughly seen as a simplification of both PoS tags and dependency relations in DS's. See also section 3.2.

[4]Tesnière uses the junction operation to represent coordinated structures and other particular joined structures, such as the apposition (e.g., *[the US president], [Obama]*). Although our implementation includes all types of junction, in this paper we will particularly focus on *coordination* since it constitutes the majority of the cases, and yet the most problematic ones.

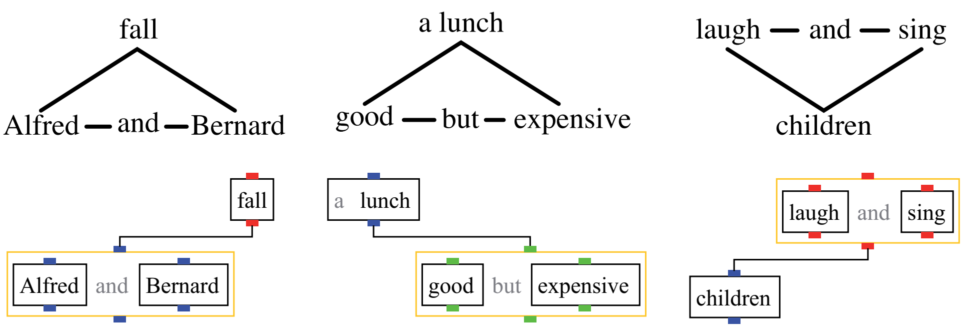


Figure 2: Examples of coordination. Tesnière's original notation is on top, and our notation at the bottom (we represent the junction with a yellow box).

2.4 Transference

The other operation is named *transference*. There are two types of transference. The first degree transference is a shifting process which makes a block change from the original category of the content word, to another[5]. This process often occurs by means of one or more empty words belonging to the same block, called *transferrers*. Figure 3 (left) shows an example of first degree transference. The word *Peter* is transferred from the word class *noun* and takes the functional label of an *adjective* via the possessive clitic *'s* which acts as a transferrer. In our representation (bottom), every block has two little colored boxes: the one at the bottom indicates the original category of the content word, and the one at the top indicates the category of the block after all transferences are applied.

The second degree transference occurs when a simple clause becomes an argument or an adjunct of another clause[6], maintaining all its previous lower connections, but changing its functional label within the main clause. The sentences below represent some examples of second degree transference:

(1) She believes *that he knows*

(2) The man *I saw yesterday* is here today

(3) You will see him *when he comes*

[5]A somehow similar operation has been formulated in the framework of Combinatory Categorial Grammar (cf. [8]), and goes under the name of type raising.

[6]In other words, the verb of the embedded clause becomes a dependent of an other verb. This should not be confused with the case of compound verbs, which are represented as a single block, where auxiliaries are labeled as empty words (see for instance the TDS in figure 1).

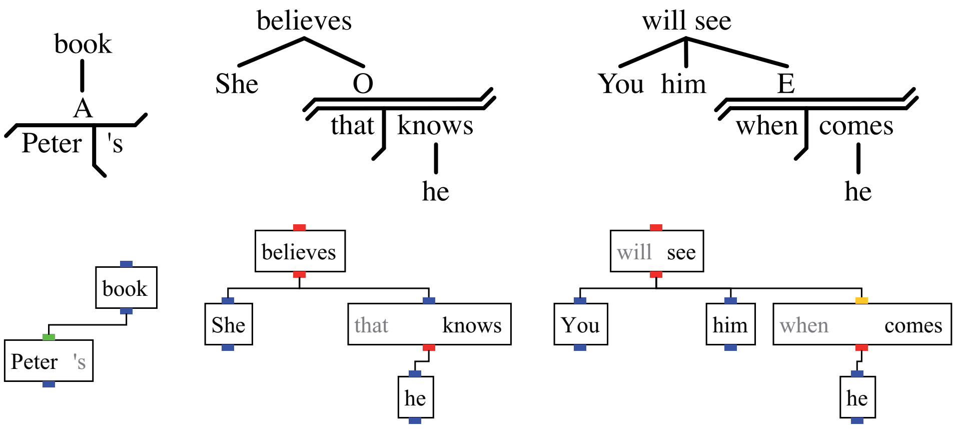


Figure 3: An example of *first degree transference* of the phrase "*Peter's book*" (left), and two examples of *second degree transference* of the sentence "*She believes that he knows*" (center) and the sentence "*You will see him when he comes*" (right).

In the first sentence, we have a transference verb$\gg$noun by means of the transferrer *that*. The embedded clause in italics takes the functional label of a noun, and becomes the object of the verb. Figure 3 (center) shows the corresponding TDS. The embedded clause in the second example has the functional label of an adjective. It is a transference verb$\gg$adjective without any transferrer. The third sentence is an example of transference verb$\gg$adverb: the clause in italics has the functional label of a temporal adverb through the transferrer *when*. Figure 3 (right) shows the corresponding TDS.

3 Advantages of TDS over DS

In this section we will describe three main advantages of using TDS notation as an alternative of currently available DS representations. In particular we will discuss the issue of choosing the linguistic heads in PS trees (section 3.1), compare how the two models categorize dependency relations (section 3.2), and how they treat the phenomenon of coordination (section 3.3).

In order to compare the different representations, Figure 4 illustrates three structures of an English sentence: the original Penn WSJ PS tree, the same structure converted to DS as in [9], and the TDS our conversion algorithm generates.

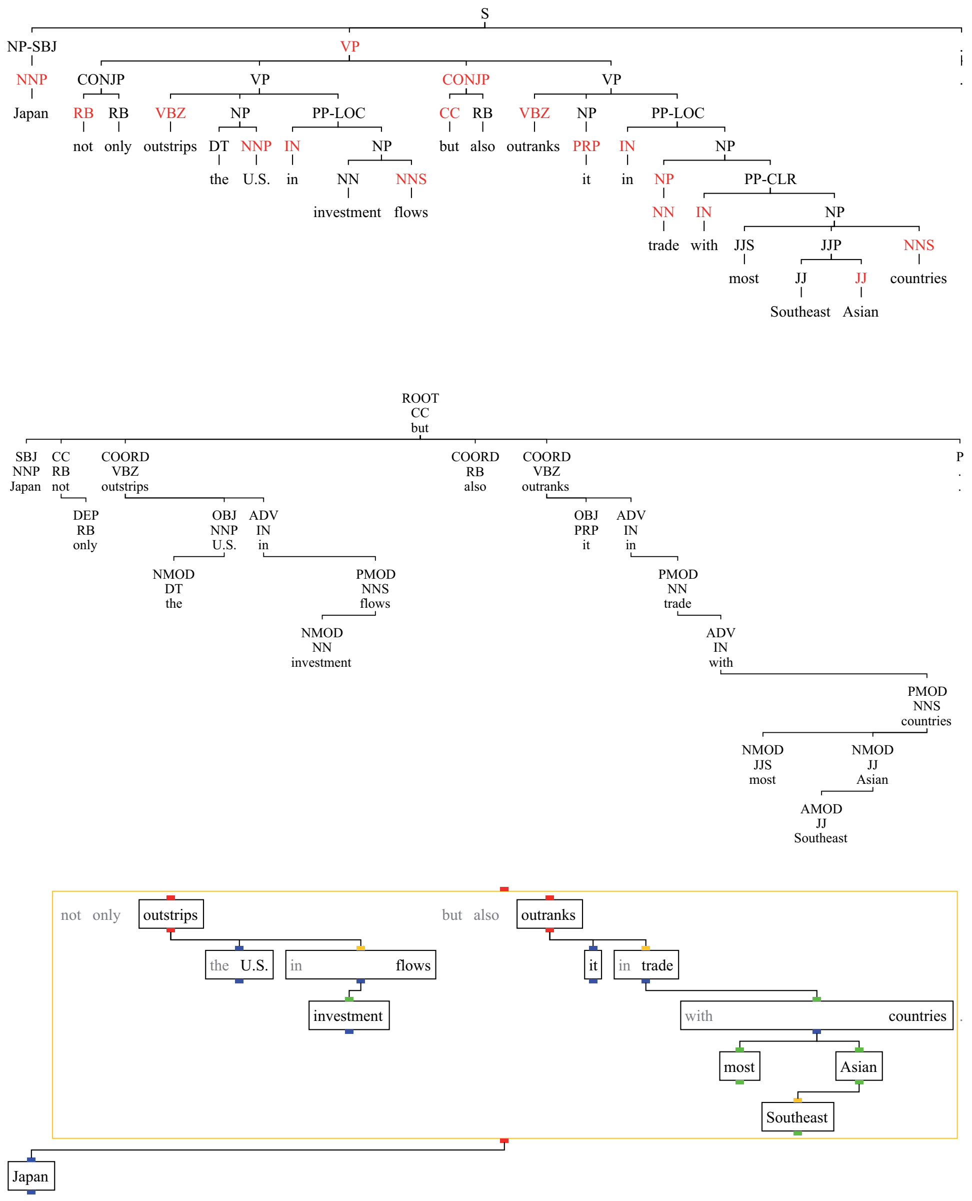


Figure 4: Comparison between different representations of an English sentence. **Top**: original WSJ PS taken from the WSJ sec-00 (#666). Null productions and traces have been removed. The red labels are the heads according to the DS below. **Center**: DS according to [9] using the `pennconverter` script in `conll2007` mode. Every word in the DS is presented together with its PoS and the label of the dependency relation with its governor. **Bottom**: TDS our conversion algorithm generates.

3.1 Choosing the correct heads

The first step in order to perform a PS-to-DS conversion is to annotate the starting PS with head labels. This procedure has been initially proposed in Natural Language Processing (NLP) by Magerman [10] and then slightly modified by others (e.g., [17], [9]). If exactly one unique head is chosen for every constituent of the PS, the enriched tree can be shown to be homomorphic to a single projective DS (cf. [7], [13]).

The choice of heads assignment is a critical one: although much linguistic literature is present on this issue (cf. [4]), in NLP there have been only few attempts to empirically evaluate different heads assignments (i.e., [2], [14]). While certain choices are less disputed (e.g., the verb is unequivocally the head of simple clauses), most of the remaining ones are contended between empty and full words. The most frequent cases are listed here:

- Determiner vs. noun in nominal phrases (e.g., *the man*).
- Preposition vs. noun in prepositional phrases (e.g., *on paper*).
- Complementizer vs. verb in sub-clauses (e.g., *I believe that it is raining*).

In TDS, all these choices become irrelevant: since every empty word is included in the block together with the content word it belongs to, no preference is needed[7].

3.2 Categories and Blocks

Currently used DS representations make use of labels to identify the dependencies between words. For example *SBJ* and *OBJ* are used to mark the relation between a verb and its subject and direct object respectively. These labels are closely related to the four categories proposed by Tesnière. The main difference is in their number: while DS uses around a dozen of different labels, TDS uses only four. This turns out to be beneficial for a more simplified and yet generalized analysis.

The other difference is more subtle. In DS every word is a node, and therefore, for every node (except for the root) we need to identify the label of the dependency relation with its governor. The problem here is related to the above discussion about the choice of heads. If we take the example in figure 3 (center), one has to choose whether the complementizer or the verb is the direct object of the main verb. TDS better represents these cases, by including both elements in the same block. This choice is justified by the fact that both elements contribute to make the node an argument or an adjunct of the verb.

[7]In TDS, heads assignment remains essential when two or more full words are sister nodes of the same constituent, such as in "the woman <u>who</u> I <u>like</u>". In this example the verb should be the head.

3.3 Coordination

Coordination represents one of the major problems in currently used DS representations (cf. [13]). If dependency[8] is the only operation available to relate words, two main strategies are adopted:

1. One conjunction (or conjunct) is the head of the other elements.
2. Each element (conjunction or conjunct) is the head of the adjacent element which follows.

The first solution is the one which is more commonly adopted in current PS-to-DS conversions. The second one is proposed by Mel'čuk in [12]. Both solutions are problematic in circumstances such as the one of figure 4. If the coordination includes multiple conjunctions, assigning the head to either one of the conjuncts or one of the conjunctions, leads to a strong asymmetry in the structure: either the conjuncts are not all at the same level, or the set of dependents includes both conjunctions and conjuncts. Moreover, if the coordination phrase is coordinating verbs at the top of the sentence structure, other potential blocks, e.g., the subject *Japan* in the example, will also appear in the set of dependents, at the same level with the verbs they depend on[9]. Finally the *conjunction phrase*, i.e., a group of words forming a single conjunction (e.g., *not only* in the example), is also poorly represented in DS representations, since it is not grouped into a unique entity.

Tesnière's choice of adding a special operation to handle coordination is justified if we consider how well it represents all the cases DS fails to represent consistently. Coordination in TDS can be seen as a borrowing of the notion of constituency from PS notation: the different blocks being conjoined have equal status, they govern all the blocks being dominated by the coordination block, and are dependents on all blocks the coordination structure depends on.

4 Converting the Penn WSJ in TDS notation

In this section we will present the current state of the project of converting the Penn WSJ treebank [11] into TDS notation. In section 4.1 we will introduce the elements composing each generated TDS, in section 4.2 we will describe the conversion procedure, and in section 4.3 we will provide some error analysis on the generated structures.

[8]We only consider the case of single headed DS, i.e., each word should have exactly one governor.

[9]The labels of the dependency relations, such as the ones in the DS of figure 4, can often help to differentiate between dependents which have the same head, but differ in their functional labels. However they cannot be considered an optimal solution, since they don't eliminate the structural asymmetry.

4.1 Elements of a TDS

Figure 5 illustrates the main elements, introduced in section 2, which we need to define in order to construct our TDS's. Words are divided into full and empty words[10], and blocks are either standard or junction blocks. A generic block contains a list of empty words, and a list of dependent blocks. In addition a standard block has to contain a unique full word, while a junction block needs to specify a list of conjunction words and a list of conjunct blocks.

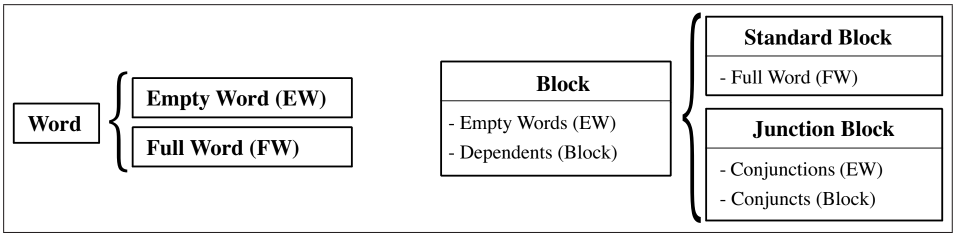


Figure 5: Word and block types.

4.2 The conversion procedure

In order to generate TDS's from the Penn WSJ, we have decided to start from the PS original annotation, instead of using already converted DS's. The main reason for this choice is that PS annotation of the WSJ is richer than currently available DS representations. This concerns in particular coordination structures, which would be much harder to reconstruct from DS notation (see section 3.3).

Each PS in the corpus is preprocessed using the procedure described in [16], in order to add a more refined bracketing structure to noun and adjectival phrases. Moreover, we remove null productions and traces from all trees, and enrich them with head labels[11].

The pseudocode reported in algorithm 1, contains the procedure which is applied to each PS of the corpus, in order to generate the respective TDS. The algorithm recursively traverses from top to bottom each node of a PS, and outputs either a junction block[12] (left) or a standard block (right).

[10]A word is empty if its PoS is one of the following: punctuation marks, CC, DT, EX, IN, MD, POS, RP, SYM, TO, WDT, WRB. Moreover special pairs of words are marked as empty (e.g., more like, more than, even though, such as, many of, most of, rather than).

[11]This operation is done using the `treep` script [2], and a rule set which is a modification of the one used in [10]. This modification mainly consists in prioritizing conjunctions' PoS within several phrase structures (e.g., NP, VP, QP). Moreover we have added few rules for the non-terminals (i.e., "NML", "JJP") introduced by the procedure described in [16].

[12]A constituent is roughly identified as a junction structure when it presents conjunctions elements (i.e., CC, CONJP), or when it is composed of subconstituents with the same labels, such as in the cases of apposition (see note 4).

```
Algorithm: Convert($N_{PS}$)
Input: A node $N_{PS}$ of a PS tree
Output: A block $N_{TDS}$ of a TDS tree
begin
    instantiate $N_{TDS}$ as a generic block
    if $N_{PS}$ is a junction then
        instantiate $N_{TDS}$ as a junction block
        foreach node D in children of $N_{PS}$ do
            if D is a conjunct then
                $D_{TDS} \leftarrow Convert(D)$
                add $D_{TDS}$ as a conjunct block in $N_{TDS}$
            else
                $D_{lex} \leftarrow$ lexical yield of D
                if $D_{lex}$ is a conjunction then
                    add $D_{lex}$ as a conjunction in $N_{TDS}$
                else
                    add $D_{lex}$ as empty word(s) in $N_{TDS}$
    else
        $N_h \leftarrow$ head daughter node of $N_{PS}$
        if $N_h$ yield only one word $w_h$ then instantiate $N_{TDS}$
        as a standard block with $w_h$ as its full word
        else $N_{TDS} \leftarrow Convert(N_h)$
        foreach node D in children of $N_{PS}$ do
            if $D == N_h$ then continue
            $D_{lex} \leftarrow$ lexical yield of D
            if $D_{lex}$ are only empty words then
                add $D_{lex}$ as empty word(s) in $N_{TDS}$
            else
                $D_{TDS} \leftarrow Convert(D)$
                add $D_{TDS}$ as a dependent of $N_{TDS}$
    return $N_{TDS}$
end
```

Algorithm 1: Pseudocode of the conversion algorithm from PS to TDS.

For each TDS the algorithm generates, several post-processing steps are applied:

1. Join together all *compound verbs* into a unique block[13].

2. Unify in a unique standard block all *contiguous proper nouns*.

3. Define the *original category*[14] of each block.

4. Define the *derived category*[15] after transferences are applied.

The conversion procedure just described has been successfully employed to generate the first TDS version of the Penn WSJ treebank. The conversion and visualization tool, together with its technical documentation, is freely available at `http://staff.science.uva.nl/~fsangati/TDS`.

[13]E.g., [is eating], [has been running]. All verbs preceding the main one, are marked as empty words. This procedure doesn't apply to junction structures, see also section 4.3.

[14]This category is specified by the PoS of its full word if it is a standard block, and by the original category of the first conjunct block, if it is a junction structure.

[15]This category is specified by the original category of the governing block (if the current block is the root of the structure the category coincides with its original category). If the governing block is a noun or an adjective, the current block is an adjective or an adverb, respectively. If the governing block is a verb, the current block is either a noun or an adverb. This last decision depends on whether the original PS node, from which the current block derives, has a circumstantial label, i.e., it contains one of the following tags: ADVP, PP, PRN, RB, RBR, RBS, ADV, BNF, CLR, DIR, EXT, LOC, MNR, PRP, TMP, VOC.

4.3 Error analysis

At this stage it is impossible to give a meaningful quantitative analysis of the overall accuracy of the conversion, since no gold corpus annotation is available for the target format. However, a manual analysis on a small sample of the corpus reveals that most of the mistakes relate to coordinated structures[16], and wrongly assigned categories (mostly arguments/adjuncts).

Moreover, in a limited number of cases, we found two possible inadequacies of the current TDS notation, when dealing with specific linguistic phenomena. The first issue concerns junction structures: while in our model the junction operation can only join blocks, several linguistic constructions show the necessity of defining it also on full (or empty) words within a block[17]. Two examples are the coordination of compound verbs (e.g., *He was [eating and drinking]*), and the coordination of empty words (e.g., *The indicator fell steadily [up to and through] the crash*).

The second case regards the transferrers: in our implementation they must be always empty words. There are however few cases in which a full words can function as a transferrer, such as *likely* in "*A forum likely to bring attention*". We will take into consideration these modifications for future updates of the model.

5 Conclusion

In this paper we have described an ongoing project of converting the Penn Wall Street Journal treebank from PS to TDS representation, inspired by the work of Tesnière [15]. Corpus-based Computational Linguistics has often valued a good compromise between adequacy and simplicity in the choice of linguistic representation. The transition from PS to DS notation has been seen as a useful simplification, but many people have argued against its adequacy in representing frequent linguistic phenomena such as coordination. The TDS conversion presented in this paper, reintroduces several key features from Tesnière's work: on one hand the operation of junction enriches the model with a more adequate system to handle conjoined structures (e.g., coordination); on the other, the blocks, the transference operation, and the category system further simplify and generalize the model.

We are currently working on a probabilistic extension of our framework. The idea is to define a language model which generates and parses sentences using the new representation. In particular, for what concerns junction structures, our intuition is that the model should generate them differently with respect to standard blocks. If our intuition is correct, the new probabilistic model could be better at modeling and predicting language structures.

[16]In certain complex coordinated structures, where conjuncts and modifiers of the coordination are put at the same level, dependent blocks are wrongly identified as conjuncts. In other cases the coordination is not detected because conjunction words are missing.

[17]This means that we would need to define an other element, in addition to the word-types of figure 5 (left), which we could call *junction word*. This new entity would have to specify a list of full (or empty) *conjunct words* and a list of (empty) *conjunction words*.

Acknowledgments

The authors are particularly thankful to Igor Mel'čuk, Willem Zuidema, Rens Bod, Yoav Seginer, Sophie Arnoult, and three anonymous reviewers for their valuable comments. FS gratefully acknowledge funding by the NWO (grant 277.70.006).

References

[1] Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. Building a Treebank for Italian: a Data-driven Annotation Schema. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 99–105, 2000.

[2] David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *Proceedings of the 19th international conf. on Comput. Linguist.* , pages 1–7, Morristown, NJ, USA, 2002.

[3] Noam Chomsky. *Syntactic structures*. Mouton, Den Haag, 1957.

[4] Greville G. Corbett, Norman M. Fraser, and Scott McGlashan, editors. *Heads in Grammatical Theory*. Cambridge University Press, New York, 2006.

[5] Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, Valia Kordoni. Towards a dependency-based gold standard for German parsers - The TiGer Dependency Bank, 2004.

[6] Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová Hladká. Prague Dependency Treebank 1.0, 2001.

[7] David G. Hays. Grouping and dependency theory. In *National Symposium on Machine Translation*, pages 258–266, Englewood Cliffs, NY, USA, 1960.

[8] Julia Hockenmaier and Mark Steedman. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Comput. Linguist.* , 33(3):355–396, 2007.

[9] Richard Johansson and Pierre Nugues. Extended Constituent-to-Dependency Conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 2007.

[10] David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.

[11] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* , 19(2):313–330, 1993.

[12] Igor Mel'čuk. *Dependency Syntax: Theory and Practice*. State Univ. of New York Press, 1988.

[13] Joakim Nivre. *Inductive Dependency Parsing (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[14] Federico Sangati and Willem Zuidema. Unsupervised Methods for Head Assignments. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 701–709, Athens, Greece, March 2009.

[15] Lucien Tesnière. *Eléments de syntaxe structurale*. Editions Klincksieck, Paris, 1959.

[16] David Vadas and James Curran. Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Comput. Linguist.* , pages 240–247, Prague, Czech Republic, June 2007.

[17] Hiroyasu Yamada and Yuji Matsumoto. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206, 2003.

Towards English-Czech Parallel Valency Lexicon via Treebank Examples

Jana Šindlerová and Ondřej Bojar

Charles University in Prague
Institute of Formal and Applied Linguistics (ÚFAL)
E-mail: {sindlerova,bojar}@ufal.mff.cuni.cz

Abstract

The paper describes an ongoing project of building a bilingual valency lexicon in the framework of Functional Generative Description. The bilingual lexicon is designed as a result of interlinking frames and frame elements of two already existing valency lexicons.

First, we give an overall account of the character of the lexicons to be linked, second, the process of frame linking is explained, and third, a case study is presented to exemplify what the information contained in frame links tells us about crosslinguistic differences in general and the linguistic theory applied.

1 Introduction

Bilingual and multilingual lexicons have been arising quite commonly on the computational linguistics scene in the past decade. Besides the simple fact that electronic bi- and multilingual dictionaries are necessary tools for NLP projects concerned with machine translation, there is also a strong assumption that bi- and multilingual valency lexicons can be useful in the MT area as well.

Many researchers in the field of lexicography underline the need to support the lexicographic data with the evidence from linguistic corpora, e.g. [2]. Dictionaries and valency lexicons thus often arise directly in the process of corpora annotation (see e.g. [4]).

Contrary to the idea of building a valency lexicon as a resource for treebank annotation, we present an ongoing project of building a bilingual valency lexicon using a treebank as an annotation tool. The project takes advantage of two already existing valency lexicons: PDT-VALLEX and Engvallex, which have been developed during the annotation of the Prague Dependency Treebank [3] and Prague Czech-English Dependency Treebank (PCEDT, [1]), and of the parallel treebank PCEDT itself.

PCEDT is a syntactically annotated parallel corpus of approximately 50,000 sentences originally from the Penn Treebank (Wall Street Journal section), translated into Czech. The merit of PCEDT lies in the fact that the core annotation takes place on the tectogrammatical layer (t-layer), i.e. on the layer of deep syntactic relations with an overrun into the area of semantic relations. The deep syntactic annotation of PCEDT is still in progress. The annotation works on the Czech part (PCEDT_CZ) and the English part (PEDT) proceed independently albeit synchronized. Currently, about 11,500 mutually corresponding sentences are finished, which amounts to about 23% of the whole corpus (though the percentage of sentences already finished on each individual side of PCEDT reaches higher, to about 40% and 60%). By the time our multilingual valency lexicon is concluded, we expect the PCEDT corpus to have been completed.

By creating a bilingual valency lexicon, we hope to gain a multifunctional resource useful in many areas. First it will provide linguistic information about the behaviour of verbal valencies in a crosslinguistic perspective. Second, the resulting multilingual valency lexicon created in a specific linguistic framework (FGD) may serve as an interesting test for the usability and appropriateness of the framework itself. Fourth, since there is an assumption of a certain degree of universal behaviour across languages, comparing the frames in the two lexicons can be used as a test of the accuracy of the lexicons. And last but not least, with respect to the fact that verbal valencies serve as the core of syntactic structure in most languages, it will provide an interesting resource for MT applications.

2 Lexicographic Process in a Parallel Treebank

2.1 Construction of Source Valency Lexicons

PDT-VALLEX[1] has been developed as a resource for valency annotation in a large-scale syntactically annotated corpus, the Prague Dependency Treebank. Information about verbal valency is embedded into the tectogrammatical layer of annotation, i.e. the layer of deep syntactic dependency relations, therefore it does not specify any surface requirements but rather syntactico-semantic requirements of the verbs. Each headword contains one or more valency frames corresponding (mostly) to the individual senses of the headword. Valency frames contain participant slots represented by tectogrammatical functors, i.e. labels from the layer of syntactico-semantic representation. Only the so called "inner participants" and obligatory "free modifiers" are included in the frame, information about possible typical background elements is not stored except for some short notes in the example area. Each slot is marked as obligatory or optional.

[1]Not to be confused with VALLEX [5], a general lexicon with very similar formal background but not tailored to any corpus.

By now, PDT-VALLEX contains 10,593 valency frames for 6,667 verbs. The verbs and frames come mostly from the data appearing in the PDT, version 2.0, the lexicon is being constantly enlarged by data gained from PCEDT annotation.

The origin of Engvallex is different, though the motivation (gaining a resource for syntactic annotation of a treebank) is similar. At the time PCEDT begun to be annotated on the tectogrammatical layer, a reliable version of PDT-VALLEX had already been finished, fully checked and published. A similar resource was needed to be available for the English annotation in a reasonable time, therefore, instead of creating Engvallex manually on the basis of pure data, we decided to adapt an already existing resource of English verbs valency characteristics, the PropBank [7].

The PropBank lexicon has been adapted to the Functional Generative Description scheme in several ways. First, all slots have been renamed using functors, second, the non-obligatory (according to FGD) free modifiers have been deleted and optional elements marked. Third, frames corresponding to the same verb sense have been merged. Fourth, the lexicon has been refined in the process of treebank annotation by addition of other frames, whole verb lemmas, and also, the PropBank adapted frames were corrected manually with respect to the language data available in the PCEDT corpus.

Engvallex only contains verbs so far. Currently, it contains 6,213 valency frames for 3,823 verbs. As in case of PDT-VALLEX, it is being constantly expanded and refined in the course of PCEDT annotation.

In the process of PropBank adaptation to FGD theory, some core differences of the two valency theories came alight that are supposed to affect also the intended linking process. For example, it appeared that the PropBank argument range is much broader than the one usually admitted by FGD approach. It results from stricter criteria for "argumentness" used in FGD (the famous *dialogue test* [8], disallowance of non-obligatory free modifiers in verb frames etc.). Such frame arguments were usually deleted during the adaptation process, though we kept them in sporadic cases where the resulting frame would have been otherwise divested of adjuncts too typical. The deleted frame arguments were typically benefactives, non-obligatory attributes of arguments, commitatives or locatives.

2.2 Annotating Types While Seeing Tokens

Our aim of aligning two existing valency lexicons is considerably easier than the lexicographic process carried out at the time the individual lexicons were built. Still, we face the problem of formally describing verb (or frame) *types* while observable items are verb *tokens*.

Traditionally, lexicographers collected corpus evidence, organized the tokens into groups of examples with similar syntactic and/or semantic properties and derived a single description of the given type. Little or no effort was spent in checking whether the description well matches the "training" tokens or even an independent set of "test" tokens. In our opinion, this is the root of troubles faced when trying to apply a traditional lexicon in NLP applications. Fortunately, recent projects (e.g. FrameNet [9]) try hard to provide enough real-world example sentences coupled with lexicon entries.

We design our annotation process to carefully separate the annotation attributed to types (i.e. lexicon entries) from the annotation attributed to tokens (i.e. verb occurrences in a treebank), but we require the annotator to see and provide both annotations simultaneously. In order to simplify the annotation process, we implement automatic procedures to project type annotation to an observed token and vice versa.

With the automatic procedures at hand, the annotator usually constructs the type annotation at the first token of the given type. Subsequent tokens of the same type will automatically reveal how the type annotation projects in the particular case. We can easily highlight any conflicts between the projection and the token annotation.

We feel that this design of the lexicographic process has several advantages:

- The annotation of types is presented not in an abstract form but rather naturally projected on a given example, i.e. verb frames are displayed in example sentences, but not available in the form of written lists of slots.
- While building the lexicon, we get an annotated corpus as a by-product, including explicit links between the two resources.
- Automatic highlighting of conflicts between the annotation of tokens and the projected annotation of types serves as quality assurance for all three components in question: the lexicon, the corpus of lexicon examples and the automatic procedures that apply lexicon entries to (unseen) sentences.

We believe that this explicit type-token link is vital for future applicability of the constructed resource. For instance, if a lexicon entry is doubted by a human user, he or she can use the treebank examples to understand better the generalization captured in the lexicon. For NLP tools, the set of annotated examples can serve as a test set or as a training set for machine-learning algorithms.

2.3 Description of Annotation Environment

2.3.1 Tools Used

The annotation tool we developed builds on two large software projects: TectoMT [11] framework for various NLP tasks (including MT) and tree editor TrEd[2].

[2] http://ufal.mff.cuni.cz/~pajas/tred

TectoMT is a modular programming environment aimed at linguistically rich processing of text. The two features of TectoMT we exploit are: automatic alignment of Czech and English t-nodes [6] and TectoMT native file format TMT, capable of storing dependency analyses of two languages at all three layers of linguistic description. As our source examples of verb usage are already manually annotated at the t-layer for both English and Czech, we do not need the automatic analyses implemented in TectoMT, but this option would be clearly very useful for a potential future annotation of a different text type.[3]

TrEd is a highly customizable and extensible editor of dependency trees. TrEd was used for manual annotation of all the above-mentioned Prague treebanks and an extension of TrEd allows to edit TMT files, i.e. to work with several trees of a given sentence pair at once.

2.3.2 Design of User Environment

The design of the user environment for the annotation follows the principles outlined in Section 2.2. The user is presented with a pair of t-trees and aligned verbs. We use several types of arrows to indicate *token annotation*, i.e. links between dependents of the verbs in this particular sentence, and *type annotation*, i.e. links indicating the correspondence between the slots of the two frames in question, see Figure 1.

The user environment facilitates the following annotation actions:

Token annotation: Correction of automatic node alignment.

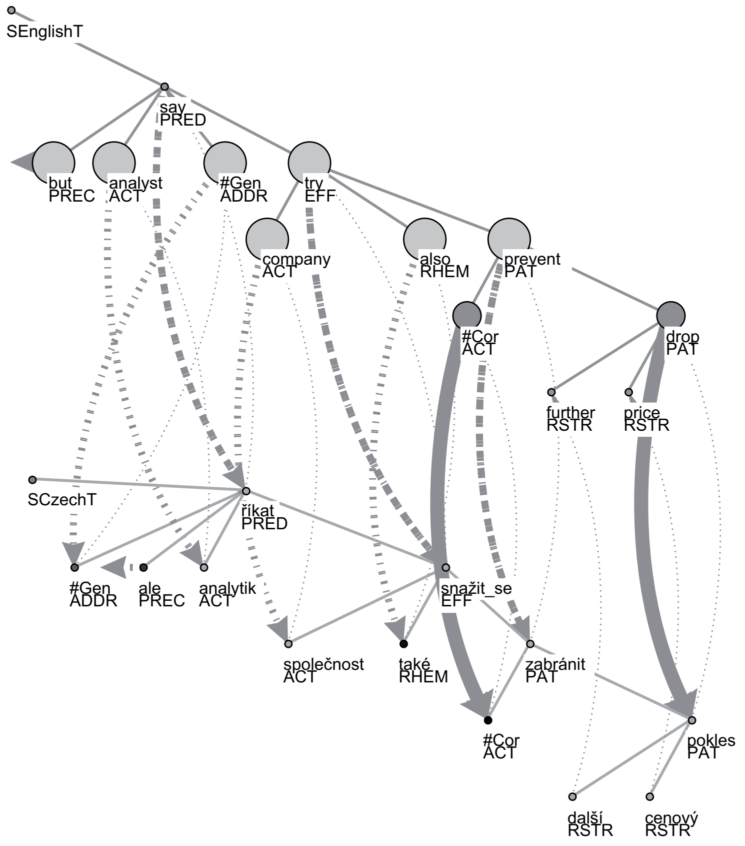
The pair of t-trees has already been automatically node-aligned, so most English t-nodes have a Czech t-node counterpart (one, at most). We use the node alignment to find both: pairs of matching verbs as well as pairs of verbs' immediate dependents.

If the automatic alignment does not provide a link, or there is an error in the alignment, the user can provide manual node alignment links simply by dragging an English node onto a Czech node. If a node is aligned both manually and automatically, the manual alignment takes precedence and the automatic alignment is not displayed at all.

Type annotation: Collection of node alignment to Engvallex.

When the manual or automatic node alignment correctly represents the alignment of verb dependents, the annotator uses a single keystroke command to collect and store it as the slot alignment in the correspoding Engvallex entry.

[3]The only step performed in our source manual trees with no automatic counterpart in TectoMT is the selection of frame ID of a given verb occurrence, i.e. the verb-frame disambiguation task. However, the task itself has already been explored for Czech [10].



But analysts say the company is also trying to prevent further price drops.
Ale analytici říkají, že společnost se také snaží zabránit dalším cenovým poklesům.

Figure 1: Sample pair of sentences with manual and automatic alignment of verb dependents and projected alignment of frame slots (thick arrows). In practice, the arrows are color-coded.

A feedback to the user visually combines both type and token annotation. For every pair of aligned verbs (indicated by dashed green arrows) we highlight immediate dependents and their alignment:

- **Manual and automatic node alignments** are displayed as dotted red and blue arrows. (The complete automatic node alignment is indicated by very thin dotted lines.)
- If the frame entry contains a **slot alignment** specification, the slot alignment is projected on the pair of verbs and indicated by thick green arrows.
- All English verb dependents with a missing or mismatching slot alignment are displayed as large (yellow) nodes.

- When the type and token alignment matches (as illustrated for the verb *prevent–zabránit* in Figure 1), the nodes are smaller (and green).

In order to simplify the access to individual verb examples, we use TrEd "filelists". A filelist contains a list of corpus positions, i.e. filenames and node IDs. Filelists allow to browse the parallel treebank data in various ways. For the time being, we prepared a filelist for each pair <English verb, its Czech translation> and we organize the filelists based on the number of corpus examples. With a different filelist, the same corpus could be browsed from the most complex verb frames or from frames with most conflicts in the (automatic) token and type annotation.

2.3.3 Implementation Details

Both parts of the parallel treebank we build upon use their respective file formats to store Czech and English t-layers. We identify sections of data annotated in both PEDT and PCEDT_CZ and merge them into TMT files. In the subsequent annotation, we use only the combined TMT files and never modify the original independent treebank files.[4]

Engvallex and PDT-VALLEX are stored in XML files with a similar but not exactly identical structure. Both lexicons are still under development. In order to avoid conflicts, we detach from their development and preserve some fixed versions of the lexicons for our purposes.[5]

Technically, manual node alignments are stored directly in the TMT files. The slot alignment information should belong to both valency dictionaries, but for the time being we prefer to store it in Engvallex only.

We extend the representation of Engvallex to include a set of frame counterparts for each frame of an English verb. Each of the frame counterparts specifies the ID of the target frame in PDT-VALLEX accompanied by a mapping of slots. As slots in both valency dictionaries are uniquely identified by their functors, the mapping simply consists of tuples <Czech slot functor, English slot functor>. The format currently permits also 1-0 mapping (no counterpart slot in the Czech frame) and we will soon also store the list of unaligned English slots, i.e. 0-1 mapping, to differentiate between no mapping and still unspecified mapping in the representation.

[4]We preserve sentence and node IDs (and do not modify the t-layer annotation apart from a few corrections in functor values), so all our annotations can be transfered back to the treebanks, if desired.

[5]Frame IDs are usually preserved, so later our alignment should be easily transferable to fresh versions of the lexicons.

3 Preliminary Observations

The most frequent problem with the annotation environment is the lack of support for coordinated verb dependents or, even worse, coordinated verbs. While this limitation does not completely block the annotation process (all problematic examples can be simply skipped), it requires the annotator to walk the filelist searching for a suitable example. The solution for examples with coordinated verb dependents is rather simple: a conjunction node should serve as a representative for both coordinated members and it should be understood as bearing the functor common to the dependents instead of a technical functor CONJ.

Another issue is caused by ellipsis: many examples do not contain dependents to fill all the slots of a frame. Currently, the annotator has to wait for an example explicitly mentioning a dependent of a given functor to be able to annotate the slot link. We plan to add artificial nodes for all slots not expressed at the t-layer so that the annotator would be able to align them.

The last issue is less important in our case but should be taken as a caveat for similar annotation enterprises. In our case, each example is a pair of t-trees, occupying a large portion of screen and requiring a short but observable time to render. If the lexicographer should be provided with many examples at once, e.g. for the purposes of comparison, the t-layer would be a too rich representation.

4 Case Study: Verbs of Commercial Transaction

Verbs of Commercial Transaction are due to the character of the corpus (WSJ texts, economic focus) one of the most common verb classes in PCEDT. What is more, they are characteristic by a great number of hypothetic arguments, which often fail to be realized in a surface syntactic structure. As such, they represent a verb class highly attractive as verbal valency investigation issue.

Due to the lack of space we will limit our observations to one member of the class only, the verb *sell.*

4.1 Sell

Sell is a typical representative of the verb class in question. The representations of its valency frames in the individual valency lexicons are in Table 1.

PropBank provides a single verb meaning with a single set of participants exemplified by several surface argument layouts. Engvallex, on the other hand, provides three different frames, though representing the same meaning of the verb (which is quite an unusual situation in FGD framework). Those three frames are exemplified further in (1)–(3) respectively.

Propbank entry:	**Engvallex entries:**			**PDT-VALLEX entry for** ***prodat/prodávat*****:**
Arg0: Seller	ACT	ACT		ACT
Arg1: Thing Sold	PAT	PAT	ACT	PAT
Arg2: Buyer	?ADDR			ADDR
Arg3: Price Paid	(EXT)	(EXT)	(EXT)	(EXT)
Arg4: Benefactive	—			—

Table 1: Comparison of valency entries for *sell–prodat/prodávat.* The question mark "?" sign indicates that the frame element is optional only and the brackets "(. . .)" around the functor label represent the information that the element is considered a free modifier, and as such it is not included in the frame.

(1) a. At last count, Candela had sold $4 million of its medical devices in Japan.
 b. Celkem prodala Candela v Japonsku své lékařské přístroje za 4 miliony dolarů.

Example (1) is an instance of the most common positive sentence constellation of the arguments. It can be seen that the three lexicons do not substantially differ in how they capture the valency properties of such uses of the verb.[6]

(2) a. A more recent novel , "Norwegian Wood" (every Japanese under 40 seems to be fluent in Beatles lyrics), has sold more than four million copies since Kodansha published it in 1987.
 b. Novějšího románu "Norské dřevo" (snad každý Japonec pod 40 zná texty Beatles) se prodalo od jeho vydání v nakladatelství Kodansha roku 1987 více než čtyři miliony výtisků.
 c. Four million copies of a more recent novel, "Norwegian Wood" . . . , have been sold since Kadansha published it in 1987.

Example (2) is an instance of a nonstandard shift of arguments (as PropBank interprets it). The sold item moves into the position of seller and its place is taken by an expression of transaction proportion. The Czech translation (2b) uses correctly a type of passive voice, which does not require an additional valency frame. Nevertheless, if we were to consider (2b) an instance of the frame constellation used in (1), we would have to consider an underlying structure such as (2c), which would have exactly the same tree representation as (2b).

Nevertheless, there is no imaginable way of justifying a transformation of this kind, for there is (almost for sure) no leading case for such a shift between a deep representation and a surface structure in the whole treebank. For this

[6]We decided to keep ADDRessee optional due to the dialogue test [8] results and due to the fact that in vast majority of examples in PCEDT it is semantically suppressed and not realized on the surface.

reason, we decided to keep a separate frame (despite the fact that the same verb sense is employed). Keeping a separate frame in this case also minimizes the risk of linkage conflict in the bilingual valency lexicon.

(3) a. At Christie's, a folio of 21 prints from Alfred Stieglitz's "Equivalents" series sold for $396,000, a single-lot record.

b. Na Christie's bylo 21 fotografií ve foliovém formátu z řady "Ekvivalenty" od Alfreda Stieglitze prodáno za 396 000 dolarů, rekordní částku za jedinou položku.

For similar reasons, we decided to keep a separate frame for (3a), though the construction evokes alternations of the type which is considered a mere derivation of the basic frame in the FGD application to English data.

Another issue connected to the verb *sell* is the issue of the element named EXT and standing for the *price* of the goods in the commercial transaction. FGD considers EXT a free modifier, not allowing it any role in the valency frame. Though with other verbs of commercial transaction, such as *pay*, the *price* argument has its place in the frame (being considered obligatory), here it falls out. This is a disadvantageous property of the linguistic framework we use and it, of course, has limiting consequences for the task of our interlinking the frame elements.

5 Conclusion and Future Work

Despite the fact that the process of our creating a bilingual valency lexicon is still at its beginnings, we have, thanks to it, already gained some important insight into the theoretical issues of crosslinguistic comparison of verbal valencies. By accessing the linguistic core of verbal valency via its treebank manifestations we are in hope of gaining a valuable, reliable and useful resource of linguistic information. The methodological solution we have chosen turns out as easy, user-friendly and effective in practical use.

We expect to continue annotation works and complete the linking process in approximately a year horizon. Further, we plan to utilize the bilingual valency lexicon in a forthcoming linguistic research in verbal valency and its impact on the verb semantic classes, and also, we would like to use the lexicon in future MT experiments.

6 Acknowledgment

The work on this project was supported by the following grants: GAUK 19008/2008, GAUK 52408/2008, MSM 0021620838, and FP7-ICT-2007-3-231720 (EuroMatrix Plus). We are grateful to Josef Toman for significant help with the implementation of our extension to TrEd and to Miroslav Týnovský for preliminary merging of PCEDT and PDT files into the TMT format.

References

[1] Jan Cuřín, Martin Čmejrek, Jiří Havelka, Jan Hajič, Vladislav Kuboň, and Zdeněk Žabokrtský. Prague Czech-English Dependency Treebank, Version 1.0. Linguistics Data Consortium, LDC2004T25, 2004.

[2] Jan Hajič and Zdeňka Urešová. Linguistic annotation: from links to cross-layer lexicons. In *Proc. of TLT 2*, pages 69–80, 2003.

[3] Jan Hajič. Complex Corpus Annotation: The Prague Dependency Treebank. In *Insight into the Slovak and Czech Corpus Linguistics*, pages 54–73, 2006.

[4] Erhard W. Hinrichs and Heike Telljohann. Constructing a Valence Lexicon for a Treebank of German. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories*, pages 41–52, 2009.

[5] Markéta Lopatková, Zdeněk Žabokrtský, and Václava Kettnerová. *Valenční slovník českých sloves*. Nakladatelství Karolinum, Praha, 2008.

[6] David Mareček, Zdeněk Žabokrtský, and Václav Novák. Automatic Alignment of Czech and English Deep Syntactic Dependency Trees. In *Proc. of EAMT 2008*, Hamburg, Germany, 2008.

[7] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, 2005.

[8] Jarmila Panevová. *Formy a funkce ve stavbě české věty [Forms and functions in the structure of the Czech sentence]*. Academia, Prague, Czech Republic, 1980.

[9] Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. FrameNet II: Extended Theory and Practice. Technical report, ICSI, 2005. `http://framenet.icsi.berkeley.edu/book/book.pdf`.

[10] Jiří Semecký. *Verb Valency Frames Disambiguation*. PhD thesis, Charles University, Prague, 2007.

[11] Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly Modular Hybrid MT System with Tectogrammatics Used as Transfer Layer. In *Proc. of ACL Workshop on Statistical Machine Translation*, pages 167–170, 2008.

Building a Large Machine-Aligned Parallel Treebank

Jörg Tiedemann
Department of Linguistics and Philology
Uppsala University, Uppsala/Sweden
jorg.tiedemann@lingfil.uu.se

Gideon Kotzé
Alpha-Informatica
Rijksuniversiteit Groningen, Groningen, The Netherlands
g.j.kotze@rug.nl

Abstract

This paper reports on-going work on building a large automatically tree-aligned parallel treebank in the context of a syntax-based machine translation (MT) approach. For this we develop a discriminative tree aligner based on a log-linear model with a rich feature set. We incorporate various language-independent and language-specific features taking advantage of existing tools and annotation. Our initial experiments on a small hand-aligned treebank show promising results even with small amounts of training data. The performance of our approach is well above unsupervised techniques reported elsewhere. This enables us to quickly create training material and alignment models for additional language pairs. In recent work, we aligned more than one million sentence pairs and started our experiments with the extraction of transfer knowledge for our example-based machine translation system.

1 Introduction

A parallel treebank consists of a collection of sentence pairs that have been grammatically tagged, syntactically annotated and aligned on sub-sentential level [15]. This alignment usually includes more than the alignment of words but also links between all the corresponding constituents of given sentence pairs. Parallel treebanks are valuable resources for a wide variety of applications. They are not only useful as a resource for the induction of translation knowledge for machine translation systems but also as a source for cross-lingual corpus-based studies, for example, in contrastive linguistics. However, our main concern is the use of parallel treebanks in the context of machine translation (MT). Since well-aligned treebanks will play a substantial role in our MT model, finding an optimal solution to the problem of tree alignment is very important. Various projects have been initiated to create aligned parallel treebanks [7, 1, 5] and most of them are based on tedious manual labor. There are some approaches to automatic tree alignment facilitating syntax-oriented translation models [16, 4, 6, 17, 10]. In contrast to these unsupervised or hand-crafted alignment systems, we propose a classifier-based alignment approach based on supervised learning.

In the following section we outline our model including a discussion of features used in our experiments. Thereafter, we show our results on the development corpus and, finally, describe on-going work of applying our model to large scale data sets.

2 Discriminative Tree Alignment

In our approach we look at the alignment of nodes in phrase-structure tree pairs. We model this alignment using a discriminative feature-based model, which has the advantage that we do not have to “invent” a generative story to explain the alignment. Hence, we use a binary classifier-based approach using a probabilistic model directly predicting the likelihood of a link a_{ij} between two nodes s_i and t_j given the features associated with these nodes. For simplicity we leave the structural nature of the tree alignment problem aside for now. We will later discuss issues of structured prediction and how we address link dependencies in our approach. In the next section, we introduce the general model applied in the tree aligner.

2.1 A Log-Linear Link Prediction Model

Similar to related work on discriminative word alignment we base our model on association features extracted for each possible alignment candidate. For tree alignment, each pair of nodes $\langle s_i, t_j \rangle$ from the source and the target language parse tree is considered and a score x_{ij} is computed that represents the degree to which both nodes should be aligned according to their features $f_k(s_i, t_j, a_{ij})$ and corresponding weights λ_k derived from training data. In our approach we use conditional likelihood using a log-linear model for estimating these values:

$$P(a_{ij}|s_i, t_j) = \frac{1}{Z(s_i, t_j)} exp\left(\sum_k \lambda_k f_k(s_i, t_j, a_{ij})\right)$$

Here, the mapping of data points to features is user provided (see section 2.3) and the corresponding weights are learned from aligned training data. As mentioned earlier, we simplify the problem by predicting individual alignment points for each candidate pair instead of aiming at structured approaches. Hence, we can train our conditional model as a standard binary classification problem. Note that contextual features can easily be integrated even though first-order dependencies on surrounding alignments are not explicitly part of the model. More details will be given below in sections 2.4 and 2.6.

In our experiments we will use a standard maximum entropy classifier using the log-linear model as stated above. One of the advantages of maximum entropy classifiers is the flexibility of choosing features. No independence assumptions have to be made and state-of-the art toolboxes are available with efficient learning strategies. In this work, we apply the freely available toolbox Megam [2].

2.2 Structured Prediction

Tree alignment is a typical structured prediction problem. Treating links in isolation as outlined above causes a lot of errors due to the existing dependencies between link decisions. Therefore, applying a binary classifier alone and linking according to the individual decisions is not a good idea. However, building a discriminative predictor for the entire structure is not feasible due to the sparsity of the data. Structured prediction is an active research area and various approaches have been proposed incorporating various dependencies and standard machine learning techniques. In our implementation we opt for a *recurrent architecture* [3] using history-based features and a sequential classification process. We experimented with two different directions: top-down and bottom-up link classification where the latter gave us better results. Therefore, we will only report these results.

The idea behind this strategy is simple: previous decisions of the global classifier are used as input features for coming decisions. Training is simple as link decisions are readily available and the classifier can learn directly using those features. In classification, we have to use a sequential setup in order to obtain history features. In the bottom-up strategy we start with predicting links for leaf nodes moving up towards the tree root. Here, we assign history features to be taken from the child nodes.

Another way of incorporating structural dependencies in prediction is to use a simple greedy alignment strategy. In tree alignment, it is common to restrict the process to one link per node. Therefore, we can define a greedy best-first alignment strategy to account for competition between link candidates [12]. Further constraints can be applied to guide the alignment even further. For example, well-formedness criteria can be defined to reject certain links [17]. These constraints basically add the following restriction: Descendants/ancestors of a source linked node may only be linked to descendants/ancestors of its target linked counterpart. We apply both, a greedy best-first strategy and well-formedness constraints. In addition, we also introduce a constraint to restrict alignments in such a way that non-terminal nodes are aligned to non-terminal nodes and terminal nodes to terminal nodes only.

2.3 Basic Alignment Features

Log-linear models are very flexible with regard to the feature functions that can be used. Any real-valued function can be used without considering their dependencies with each other.

Zhechev & Way [17] introduce lexical probabilities to be used in unsupervised tree alignment. They are combined into an alignment score γ which is composed out of so-called *inside* scores ($\alpha(s_l|t_l)$, $\alpha(t_l|s_l)$,) and *outside* scores ($\alpha(\overline{s_l}|\overline{t_l})$, $\alpha(\overline{t_l}|\overline{s_l})$). We use a slightly modified definition of inside/outside scores which involves the selection of the maximum lexical score for each token instead of an averaged sum over all possible word connections ($x_i \geq x$ denoting dominance):

$$\begin{aligned}\gamma(s_l,t_l) &= \alpha(s_l|t_l)\alpha(t_l|s_l)\alpha(\overline{s_l}|\overline{t_l})\alpha(\overline{t_l}|\overline{s_l}) \\ \alpha(x|y) &= \prod_{x_i \geq x} max_j P(x_i|y_j)\end{aligned}$$

In our experiments this modification gave us a better performance and intuitively this is also more appealing. Other features can be derived directly from an existing word alignment. We define a feature measuring the proportion of *consistent* links among all *relevant* links L_{xy} involving either source s_x or target language words t_y dominated by the current tree nodes (s_i and t_j).

$$\begin{aligned}align(s_i,t_j) &= \sum_{L_{xy}} consistent(L_{xy},s_i,t_j) / \sum_{L_{xy}} relevant(L_{xy},s_i,t_j) \\ consistent(L_{xy},s_i,t_j) &= \begin{cases} 1 & \text{if } s_x \geq s_i \wedge t_y \geq t_j \\ 0 & \text{otherwise} \end{cases} \\ relevant(L_{xy},s_i,t_j) &= \begin{cases} 1 & \text{if } s_x \geq s_i \vee t_y \geq t_j \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

Any sub-sentential alignment can be used with the definition above. In our experiments we apply the Viterbi word alignments produced by Giza++ [13] using the IBM 4 model in both directions, the union of these links and their intersection. This gives us four separate feature functions using the definition above.

Another word alignment feature is defined for pairs of terminal nodes. It is simply a binary feature being set to one if and only if both nodes are linked in the underlying word alignment. This is useful if terminal node alignment is included in the tree alignment model as it is in our initial experiments.

It is also possible to define a number of features that are independent of external tools and resources. Using the relative position of each node in the parse tree we define two features: tree-level similarity (tls) and tree span similarity (tss). For the former we use the distance to the root node ($d(s_i,s_{root})$ resp. $d(t_i,t_{root})$) and normalize this distance with the size of the tree (maximum distance between any node and the root node). For the latter we compute the relative "horizontal" position of a node based on the span ($s_{start} \rightarrow s_{end}$) of the entire subtree which is rooted in that node (s_i). This position is than normalized by the length (= number of leaf nodes) of the sentence (S). Furthermore, we define a *leafratio* feature that measures the difference in subtree spans (in terms of number of leaf nodes within that subtree). The formal definitions of these *tree features* are as follows:

$$\begin{aligned} tls(s_i,t_j) &= 1 - abs\left(\frac{d(s_i,s_{root})}{max_x d(s_x,s_{root})} - \frac{d(t_i,t_{root})}{max_x d(t_x,t_{root})}\right) \\ tss(s_i,t_j) &= 1 - abs\left(\frac{s_{start}+s_{end}}{2*length(S)} - \frac{t_{start}+t_{end}}{2*length(T)}\right) \\ leafratio(s_i,t_j) &= \frac{min(|leafnodes(s_i)|,|leafnodes(t_j)|)}{max(|leafnodes(s_i)|,|leafnodes(t_j)|)} \end{aligned}$$

Finally, we also define features derived from the annotation. Intuitively, category labels (for non-terminal nodes) and part-of-speech labels should be valuable indicators for a possible link. These features are simply binary features which are set to one if the particular label combination is present and zero otherwise.

2.4 Contextual Features

So far we only considered features directly attached to the candidate nodes. However, tree nodes are connected with other nodes in the tree structure and their alignment may as well depend on features of neighboring nodes. Therefore, features from surrounding nodes should be considered as well. Using the tree structure we can extract the same features as described above from other nodes connected to candidate nodes. For this we define the following functions that can be used to move within the tree when extracting features: *parent* - move to the immediate parent and take the feature values from this node; *child* - compute the average feature value for all child nodes; *sister* - compute the average feature value for all nodes with the same parent node. These functions can be applied recursively. For example, applying *parent* twice will force the feature extraction process to move to the grand-parent node (if this node exists). Note that these functions can be applied to either source or target language tree or both. In this way we have many possibilities to explore contextual features. Proper feature engineering is necessary to define useful templates.

2.5 Complex Features

Some features may be correlated in a non-linear way. To account for those it is possible to create complex features. Any of the features above (also contextual ones) can be combined in such a way that they form a new feature function with their values combined. In our approach we simply compute the product of the feature values. Other types of combinations might be possible as well. This gives us a combinatorial explosion of possible features and careful feature engineering is necessary again to select valuable ones. Furthermore, complex features are even more exposed to sparseness problems. Nevertheless, various combinations lead to significant improvements as we will see in our experiments.

2.6 Link Dependency Features

The last category of features refers to history features mentioned earlier in our discussion on structural prediction. In our implementation we use two types of history features: (1) The *children_links* feature is the number of links between direct child nodes of the current node pair. (2) The *subtree_links* feature is the number of links between all children of the current node pair. Both values are normalized with the maximum number of child nodes on either source or target side. In classification, only the prediction likelihood is used for estimating these feature values. In other words, we use "soft counts" instead of counting actual links. Classification can then be done in a bottom-up fashion before applying the greedy best-first search in the final step.

3 Alignment Results

During the development of our tree aligner we applied the Smultron treebank [7] for evaluating its performance with various strategies and feature sets. Smultron includes two trilingual parallel treebanks in English, Swedish and German. The corpus contains the alignment of English-Swedish and German-Swedish phrase structure trees from the first two chapters of the novel "Sophie's World" by Jostein Gaarder and from economical texts taken from three different sources. The alignment has been done manually using the Stockholm Tree Aligner [11]. The alignment includes *good* links and *fuzzy* links. We will use both but give them different weights in training (good alignments get three times the weight of fuzzy and negative examples). We mainly worked with the English-Swedish treebank of Sophie's World which includes roughly 500 sentences per language (6,671 good links and 1,141 fuzzy links). In our experiments we used the first 100 aligned parse trees for training and the remaining part for testing.

For evaluation we use the standard measures of precision, recall and balanced F-scores as they are used in word alignment evaluation. Due to the distinction between good and fuzzy alignments we compute values similar to word alignment evaluation scores in which "sure" and "possible" links are considered (S refers to the good alignments in the gold standard and P refers to the possible alignments which includes both, good and fuzzy links; A are the links proposed by the system):

$$Precision = \frac{|P \cap A|}{|A|} \qquad Recall = \frac{|S \cap A|}{|S|}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

The upper part of table 1 summarizes the results for different feature sets when running the aligner on our development corpus. The alignment results are very promising. We can see that adding features consistently helps to improve the performance. The advantage of a discriminative approach with a rich feature set can

settings	Precision	Recall	F
lexical features	64.89	51.00	57.11
+ tree features	56.80	60.70	58.68
+ alignment features	62.94	62.83	62.88
+ label features	77.20	74.44	75.79
+ context features	**79.77**	**75.66**	**77.66**
train=novel, test=economy	81.49	74.76	77.98
train=economy, test=novel	77.67	75.04	76.33

Table 1: Results for different feature sets (top) and textual domains (bottom).

be seen when comparing our results with the performance of an unsupervised tree aligner. Running the subtree aligner described in [17] on the same data set yields an F-score of 57.57% which is similar to the scores we obtained when using the same features only. When using a better model for estimating lexical probabilities (more data: Europarl+SMULTRON) the performance improves only slightly to about 58.64%. We, can see that the additional features and the optimization of their contributions through machine learning has a strong positive effect on the performance. A drawback of supervised techniques is that we have to drop the generality of the unsupervised approach and require aligned training data to build language pair specific models. Furthermore, there is a risk of overfitting. In order to test the flexibility of our approach we ran several cross-domain experiments using the two domains present in the Smultron treebank. The results are presented in the lower part of table 1. As we can see there is only a slight drop in performance when training on a different textual domain. However, we still have reasonably high accuracy which is certainly encouraging especially considering the effort of human annotation necessary when preparing appropriate training data.

Finally, we also looked at the training curves with varying amounts of training data. For this we used about a third of the corpus (2667 links) for testing and trained on parts of the remaining data. Figure 1 shows the impact of training size on F-scores for three feature settings.

We can clearly see that the aligner yields a high performance already with little amounts of training data. Depending on the features the training curve levels out already at training sizes way below 100 sentence pairs. This is especially apparent for the settings that include word alignment features. Their values bear a lot of positive link evidence on their own and corresponding weights do not need to be adjusted very much. Other features such as the binary label-pairs require larger amounts of training examples (which is not very surprising). However, the alignment performance does not seem to improve significantly after about 128 sentence pairs even with those features included (see, for example, “no wordalign features” in figure 1).

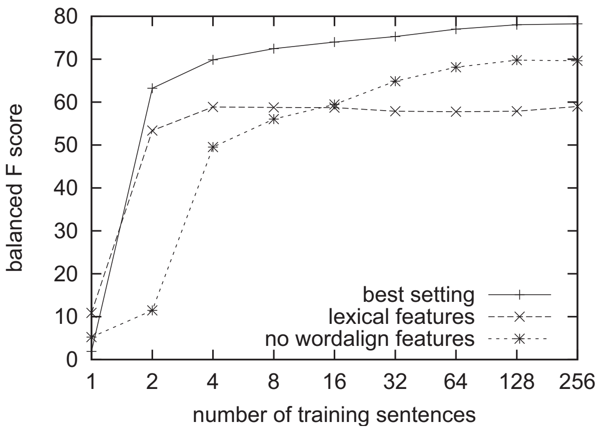


Figure 1: Training curve for various amounts of training data

4 Scaling Up

4.1 Creating a New Alignment Model

After tuning and testing our tree aligner as described above the next step is to apply the optimized model to large parallel treebanks. Unfortunately, to our knowledge no tree-aligned training corpus is available for the language pairs we are interested in (English-Dutch and French-Dutch). Therefore, the first step is to create a training corpus for our purposes. Fortunately, from our experiments we can conclude that only a small amount of aligned sentences is necessary to obtain reasonable results. Therefore, we decided to create a small aligned treebank to start with. In particular, we aligned 100 sentences from the English-Dutch parallel Europarl corpus [8] using the Stockholm Tree Aligner. We decided to follow the guidelines as proposed by [14] in the context of the Smultron parallel treebank project.

In order to speed up the process we decided to leave out all links between terminal nodes. This reduces the annotation effort dramatically not only in terms of number of links that have to be drawn but also in terms of alignment decisions. There are usually a lot of difficult decisions when aligning words with each other as we know from the literature on statistical MT and alignment competitions. A drawback is, of course, that we will miss this information when learning our alignment model. However, we believe that we will not lose a lot in performance if we take links between terminal nodes directly from existing word alignments.

For aligning the non-terminal nodes we decided to follow some guidelines as proposed by [14] in the context of the Smultron parallel treebank project:

- Align as many nodes as possible.
- They must be translatable to each other.
- Grammatical forms do not need to fit.

- Approximate translations are accorded fuzzy links. For example, one side may contain more general or more specific information.
- Pronouns are not linked to full noun phrases.
- Nodes with extra information that is missing and required on the other side are not linked. This ensures that a sentence is not linked to a verb phrase, for example.
- We only allow 1:1 links between non-terminal nodes.

We modified our implementation such that it can be run to consider only non-terminal nodes in training and alignment which is very straightforward. The software is also able to recognize existing links when aligning trees which is important when adding non-terminal links to a word-aligned parallel treebank. This is especially important for proper handling of history features and checking well-formedness criteria. Using these settings we trained a new model on our tiny set of training examples. Note that we used alignment features (for example, lexical probabilities) from a GIZA++ model trained on the entire Europarl corpus in order to get reliable estimates.

We also performed a sanity check with our newly created training corpus in order to see if the expected performance can be reached. For this we used 80 aligned sentences for training and the remaining ones for testing. This gives us a good idea about the expected outcome although the test set is way too small to give reliable evaluation scores. The results are as follows:

Precision	73.20%
Recall	86.89%
F	79.46%

As we can see, the scores are quite high especially in recall. This is partially due to the nature of our training material in which we prefer accurate links and tried to avoid fuzzy links as much as possible (in order to reduce the number of unreasonable/questionable links). Note that we only consider non-terminal nodes now in this evaluation. From this little experiment we may conclude that we can expect reasonable alignments even for our new language pair and a different textual domain.

4.2 Aligning Europarl

We are now able to align the entire Dutch-English part of the parallel Europarl corpus with the alignment model created as described above. For this we parsed both sides of the corpus with existing tools (Alpino for Dutch and the Stanford Parser for English) and converted the output of these parser to Tiger XML, which is used in our tree aligner implementation. The advantage of Tiger XML is that it supports even non-projective structures and it is also used by the Stockholm Tree Aligner which enables us to visualize alignment results easily (see figure 2).

Moreover we can even manually correct automatic alignments in this way which might be a valuable feature in future research.

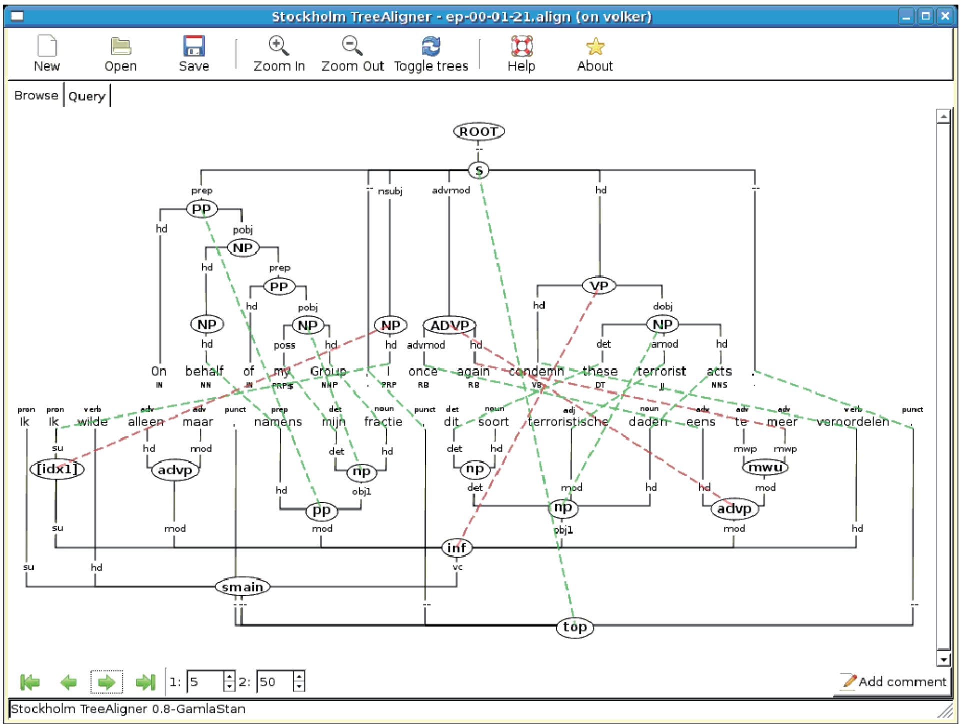

Stockholm TreeAligner - ep-00-01-21.align (on volker)
New
Open
Save
Zoom In
Zoom Out
Toggle trees
Help
About
Browse
Query
ROOT
S
On behalf of my Group I once again condemn these terrorist acts
Ik wilde alleen maar , namens mijn fractie , dit soort terroristische daden eens te meer veroordelen .
smain
inf
top
1: 5
2: 50
Add comment
Stockholm TreeAligner 0.8-GamlaStan

Figure 2: Automatic tree alignments visualized by the Stockholm Tree Aligner

After these pre-processing steps we proceeded in the following way: First we added all word alignments from the intersection of the two Viterbi alignments created by GIZA++. They were classified as "good" links assuming that they are very reliable. Next, we added links from the word alignment using the "grow-diag" symmetrization heuristics [9]. These links are less reliable and therefore were tagged as "fuzzy". Finally we ran the tree aligner to add non-terminal nodes to the parallel treebank.

There are some consequences of this alignment pipe-line architecture especially in combination with the well-formedness criteria. Word alignments often cause violations of these constraints and block the possibility of adding links on higher nodes. One solution to this problem is to disable the well-formedness check between terminal and non-terminal nodes and to allow violations at this level. However, this is not really desirable. Therefore, we use another mode in which existing word links compete with new incoming links in the same way as usual. For this we need to specify link scores for existing word alignments. In our experiments we simply defined arbitrary fixed scores (0.8 for links coming from the intersection and 0.4 for links coming from the grow-diag heuristics). As a consequence, certain word alignments may disappear from the final result if they are overruled by other stronger links that do not meet the constraints with these particular word alignments (which is probably a good thing).

Using this setup we aligned the entire Dutch-English Europarl corpus. Note that we only apply the aligner to one-to-one sentence alignments and that some sentences were dropped due to parsing time-outs. However, we obtained a substantial amount of machine-aligned parse trees (see table 2).

	good ($p > 0.5$)	fuzzy	all
Europarl English-Dutch	20,109,174	5,205,048	25,314,222
non-terminals	6,999,851	3,821,970	10,821,821
terminals	13,109,323	1,383,078	14,492,401
links/sentence	19.01	4.92	23.93

Table 2: Links in 1,057,875 aligned parse trees from Europarl Dutch-English

5 Conclusions

We have presented a discriminative tree aligner that can be trained on small amounts of hand-aligned training data using a rich feature set. With this tool we achieve state-of-the-art performance in tree-to-tree alignment that can be used to align parallel treebanks on a larger scale. We are currently aligning large automatically parsed parallel corpora which we will use as a resource in the development of a syntax-oriented data-driven machine translation system.

References

[1] Lars Ahrenberg. LinES: An English-Swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, 2007.

[2] Hal Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Implementation available at `http://hal3.name/megam/`, 2004.

[3] T.G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30. Springer, 2002.

[4] Daniel Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of ACL-03*, pages 80–87, Sapporo, Japan, 2003.

[5] Annette Rios Gonzales, Anne Göhring, and Martin Volk. A Quechua-Spanish parallel treebank. In *Proceedings of TLT7*, 2009.

[6] Declan Groves, Mary Hearne, and Andy Way. Robust sub-sentential alignment of phrase-structure trees. In *Proceedings of COLING 2004*, pages 1072–1078, Geneva, Switzerland, 2004.

[7] Sofia Gustafson-Čapková, Yvonne Samuelsson, and Martin Volk. SMULTRON (version 1.0) - The Stockholm MULtilingual parallel TReebank. http://www.ling.su.se/dali/research/smultron/index.htm, 2007. An English-German-Swedish parallel Treebank with sub-sentential alignments.

[8] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, 2005.

[9] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL-07*, Prague, Czech Republic, 2007.

[10] Alon Lavie, Alok Parlikar, and Vamshi Ambati. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation*, pages 87–95, Columbus, Ohio, 2008.

[11] Joakim Lundborg, Torsten Marek, Maël Mettler, and Martin Volk. Using the Stockholm TreeAligner. In *Proceedings of TLT6*, pages 73–78, Bergen, Norway, 2007.

[12] I. Dan Melamed. *Empirical Methods for Exploiting Parallel Texts*. The MIT Press, 2001.

[13] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[14] Yvonne Samuelsson and Martin Volk. Alignment tools for parallel treebanks. In *Proceedings of GLDV Frühjahrstagung 2007*, 2007.

[15] Yvonne Samuelsson and Martin Volk. Automatic phrase alignment: Using statistical n-gram alignment for syntactic phrase alignment. In *Proceedings of TLT6*, pages 139–150, Bergen, Norway, 2007.

[16] Wei Wang, Jin-Xia Huang, Ming Zhou, and Chang-Ning Huang. Structure alignment using bilingual chunking. In *Proceedings of COLING 2002*, pages 1–7, Taipei, Taiwan, 2002.

[17] Ventsislav Zhechev and Andy Way. Automatic generation of parallel treebanks. In *Proceedings of COLING 2008*, pages 1105–1112, 2008.

Cross-Lingual Projection of LFG F-Structures: Building an F-Structure Bank for Polish

Alina Wróblewska† and Anette Frank‡

alina.wroblewska@ipipan.waw.pl frank@cl.uni-heidelberg.de

†Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

‡Department of Computational Linguistics, Heidelberg University, Germany

Abstract

Various methods aim at overcoming the shortage of NLP resources, especially for resource-poor languages. We present a cross-lingual projection account that aims at inducing an annotated treebank to be used for parser induction for Polish. Our approach builds on Hwa et al.'s projection method [7] that we adapt to the LFG framework. The goal of the experiment is the induction of an LFG f-structure bank for Polish. The projection yields competitive results. The resulting f-structure bank may be used to train a dependency parser for Polish, or for automatic induction of a probabilistic LFG grammar.

1 Introduction

Natural language processing has made rapid progress over the last decades. Yet, computational linguistic resources and tools are restricted to a handful of languages. The creation of high-quality resources for all languages using traditional, manual techniques is a time-consuming and expensive process. This holds especially for the creation of grammars and syntactic treebanks. Various methods aim at overcoming the shortage of NLP resources (such as bootstrapping, unsupervised learning, cross-lingual projection). The approach pursued in this paper targets the induction of linguistic annotations in a cross-lingual setting: Using a bilingual corpus, existing analysis tools are applied to the resource-rich language side of a bitext. The resulting annotations are projected to the second, resource-poor language via automatically produced word alignment links. This annotation projection approach for resource induction is built on the assumption that the linguistic analysis of a sentence carries over to its translation in an aligned parallel corpus. While this assumption does not hold uniformly, the projected annotations can be used to train NLP tools for the target language (cf. Section 2).

Within the ParGram project [3], grammars for English, French, German, Norwegian, Japanese, Urdu and further languages are written according to the framework of Lexical Functional Grammar (LFG), using the Xerox Linguistic Environment (XLE) as a processing platform. Manual development of large-scale LFG grammars is an expensive process that may be sped up by automation techniques. One strand of work that targets the automatic induction of LFG grammars is the induction from existing syntactic treebanks (Cahill et al. [4]). However, this method relies on the availability of high-quality treebanks. To overcome the need of manual creation of treebanks, we investigate the cross-lingual projection approach to induce syntactically annotated corpora for new languages. Given the considerable divergence of constituent structures across languages, the grammar architecture of LFG, with its strong lexicon component and multiple levels of representations seems especially suited for a cross-lingual grammar induction task. F-structures are largely invariant across languages, and are thus especially suited to serve as the pivot for cross-lingual syntactic annotation projection. Following this insight, we pursue cross-lingual projection of grammatical functions to induce an f-structure bank for Polish. Our approach builds on Hwa et al. [7] and adapts their method to the LFG framework. We project the f-structure analyses of automatically parsed English sentences in a bitext to their Polish translations via word-alignment links. As the projected annotations are typically noisy, we apply a number of post-correction rules and filtering methods. The induced f-structure bank can be used to train a dependency parser for Polish. A full-fledged LFG grammar for Polish may be obtained by mapping the induced f-structures to appropriate c-structures (cf. Klein [8]), and using the obtained c- and f-structure bank for automatic LFG grammar induction, following the method of Cahill et al. [4].

Our paper is structured as follows. Section 2 gives an overview of the state of the art relevant to this work: We introduce the theoretical assumptions and related works on cross-lingual projection. We then focus on the projection of syntactic dependency relations and review existing computational linguistic resources for Polish. In Section 3 we outline relevant facts about the Polish language. We present the transposition of Hwa et al.'s work to the LFG framework and describe its application to the Polish language. Section 4 presents the data and experiments we conducted to induce an f-structure bank for Polish. Finally, we carry out some error analysis and compare our results to related work. Section 5 concludes.

2 State of the Art

2.1 Cross-lingual Annotation Projection

The cross-lingual annotation projection method consists in applying available monolingual NLP resources and tools in a multilingual scenario. Existing analysis tools are applied to the source language side of a word-aligned parallel corpus. Based on the assumption that the linguistic analysis of a sentence carries over to its translation in the bitext, the resulting linguistic annotations are projected from the source language onto the target language using automatic word alignment as a bridge. Annotation projection results in an automatically annotated corpus in the target language that can be used for supervised induction of NLP tools.

While the underlying assumption of cross-lingual correspondence is rather strong, the cross-lingual projection method has been successfully applied to various levels of linguistic analysis and corresponding NLP tasks, such as PoS tagging and NP bracketing (Yarowsky and Ngai [14]), syntactic dependency annotation and parser induction (Hwa et al. [7], Ozdowska [10]), argument identification (Bouma et al. [1]), word sense disambiguation (Diab and Resnik [6]), semantic role labelling (Padó and Lapata [11]) and temporal labeling (Spreyer and Frank [12]).

2.2 Projection of Syntactic Dependencies

As shown in the pioneering work of Hwa et al. [7], syntactic dependencies are especially suitable for cross-lingual projection of syntactic information, as dependency relations can carry information across languages with varying word order. Specifically, the projection of syntactic dependencies is based on the *Direct Correspondence Assumption*, which states that the dependencies in a source sentence directly map to the syntactic relationships in the word-aligned target translation:

> Given a pair of sentences E and F that are (literal) translations of each other with syntactic structures $Tree_E$ and $Tree_F$, if nodes x_E and y_E of $Tree_E$ are aligned with nodes x_F and y_F of $Tree_F$, respectively, and if syntactic relationship $R(x_E, y_E)$ holds in $Tree_E$, then $R(x_F, y_F)$ holds in $Tree_F$. Hwa et al. (2005:314) [7]

Hwa et al. [7] apply this method for annotation projection and the induction of dependency parsers for Spanish and Chinese. In their experiments, the English side of a word-aligned parallel corpus is annotated with dependency representations. These annotations are directly projected onto the target language side. Since the direct projections are noisy, the projected representations are post-processed using about 12 language-specific correction rules. The transformed representations are used to train dependency parsers for Spanish and Chinese. The results obtained by Hwa et al. [7] will be presented in Section 4, in comparison to our results.

Two further studies apply cross-lingual methods for the induction of dependency relations. Ozdowska [10] projects part-of-speech tags, morphological information (gender and number) and syntactic dependencies from two source languages (English or French) onto one target language (Polish) using only one-to-one word alignments. The aim of this experiment is to verify which source language (English or French) is more suitable for the projection of particular annotations onto Polish. Even though the annotations for both source languages are available, Ozdowska does not test whether projection from both languages in a triangulation architecture could increase the results. By contrast, the triangulation method is explored in the multi-parallel annotation projection architecture proposed by Bouma et al. [1]. Here, selected grammatical functions are projected from (one or several) source language(s) (German, English) with the aim of verb argument identification in the target language (Dutch). The approach is similar to ours, since it is based on the Pargram LFG grammars and the target of the projection are grammatical functions. However, the goal of our experiment is the induction of full-fledged f-structures, as opposed to verb-argument functions as in Bouma et al. [1]

In our work, we build on Hwa et al.'s approach that combines direct projection with language-specific post-correction rules. These rules target principled differences between the source and target languages that the DCA fails to capture, and thus offer a focused way for improving precision without impeding recall. Our account may still be complemented by triangulation techniques in later stages.

2.3 Computational Linguistic Resources for Polish

Polish is a language with relatively few NLP resources and tools. Currently, the following resources are available for Polish: corpora (e.g. the morpho-syntactically annotated corpus IPI PAN, PWN Corpus), morphological analyzers (e.g. Morfeusz, SAM, Morfologik), stemmer (Lametyzator), parsers (a DCG parser Świgra, a shallow parsing and disambiguation system Spejd)[1] and the Polish Wordnet Słowosieć.[2] Thus, there is a strong need for investigating methods for the rapid creation of further high-quality NLP resources for Polish.

3 Cross-lingual Induction of a Polish F-Structure Bank

3.1 Some Facts about Polish

In contrast to our source language English, Polish is rather unfamiliar to most readers and has been discussed and processed in few NLP studies. We briefly outline the main characteristics of the Polish syntax, with focus on syntactic phenomena that may be relevant to the projection task.

Morphology and word order. Polish is an inflecting language with relatively free word order and morphological identification of grammatical functions, by assignment of case. Thus, constituent order is rather flexible, as seen in (1.a,b).

(1)	a.	*Tomek*	*kocha*	*Marię*.
		Tom.NOM-SUBJ	love.3.SG	Mary.ACC-OBJ
		'Tom loves Mary.'		
	b.	*Marię*	*kocha*	*Tomek*.
		Mary.ACC-OBJ	love.3.SG	Tom.NOM-SUBJ
		'Tom loves Mary.'		

Pro-drop. As a pro-drop language, Polish allows omission of a personal pronoun in SUBJ function. The morphological features of the omitted subject are specified by the verb; the SUBJ is represented by an 'empty' pronoun PRED = 'pro', see (2).

[1]See overview on ACL Wiki: http://aclweb.org/aclwiki/index.php?title= Resources_for_Polish.

[2]See plWordNet Słowosieć at http://www.plwordnet.pwr.wroc.pl/browser/?lang=en

(2)

Gotuję	*obiad.*	
cook.1.SG.PRES	dinner.MASK.SG.ACC	

‘I am cooking a dinner.’

$$\left[\begin{array}{ll} \text{PRED} & \text{'gotować<SUBJ,OBJ>'} \\ \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'pro'} \end{array}\right] \\ \text{OBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'obiad'} \\ \text{CASE} & \text{ACC} \end{array}\right] \end{array}\right]$$

Null specifiers. Polish does not possess articles corresponding to ‘a/an’ and ‘the’ in English that function as SPECifiers. However, there are some pronouns that may function as determiners in SPECifier function: possessive (*mój* ‘my’, *twój* ‘your’, etc.), demonstrative (*ten* ‘this.MASK’, *ta* ‘this.FEM’, etc.), quantificational (*niektórzy* ‘some.MASK’, *wielu* ‘many.MASK’, etc.), interrogative (*jaki* ‘what.MASK’, *która* ‘which.FEM’, etc.).

Negation and case marking. **(a)** In the so-called **genitive of negation**, an accusative-marked OBJect changes to genitive case if the verb is negated, while other arguments of the verb remain unaltered. In (3.a) the verb *czytać* (engl. ‘to read’) requires an accusative OBJect argument, while under negation in (3.b), the only possible case of the OBJ argument is genitive.

(3) a.

Czytam	*ksiażkę.*
read.1.SG.PRES	book.ACC

‘I’m reading a book.’

b.

Nie	*czytam*	*książki.*
not	read.1.SG.PRES	book.GEN

‘I’m not reading any book.’

(b) A special phenomenon of case marking called **feature indeterminacy** (Dalrymple and Kaplan [5]) is observed in coordination constructions such as (4).

(4)

Kogo	*Jan*	*lubi*	*a*	*Jerzy*	*nienawidzi*?
who.ACC/GEN	John.NOM	likes.3.SG.PRES	Cnj	George.NOM	hates.3.SG.PRES

‘Who does Jan like and George hate?’

In this type of construction, two verbs with conflicting object case marking are coordinated: *lubić* (engl. ’like’) requires accusative object marking, while *nienawidzić* (engl. ’hate’) requires its object to be marked genitive. The interrogative pronoun *kogo* (engl. ‘who’) fulfills the OBJ function that is distributed over the coordinated verb predicates. It can do so because of case syncretism, i.e. because it shows the same surface form both in accusative and genitive case. Dalrymple and Kaplan [5] account for this phenomenon in a set-based theory of case-marking, by defining indeterminate (or a set of) case(s) for *kogo*, as seen in (5).

(5)

kogo	‘who’:	$(\uparrow \text{CASE}) = \{\text{ACC}, \text{GEN}\}$
lubić	‘like$\langle$SUBJ, OBJ$\rangle$’:	$\text{ACC} \in (\uparrow \text{OBJ CASE})$
nienawidzić	‘hate$\langle$SUBJ, OBJ$\rangle$’:	$\text{GEN} \in (\uparrow \text{OBJ CASE})$

Since our account will be based on the projection of grammatical functions, genitive of negation or feature indeterminacy as special cases of case marking will not constitute any problem, as the grammatical functions involved remain constant.[3] Missing subjects and specifiers constitute structural divergences at the surface level and need to be accounted for in the projection module that considers special word alignment configurations (see below, Section 3.2). The fact that Polish has a relatively free word order fits nicely with LFG's conception of f-structures and grammatical functions, which are represented independently of surface word order.

3.2 Cross-lingual Projection of LFG F-Structures

The aim of our work is to create an LFG f-structure bank for Polish with minimal human involvement. We build on Hwa et al.'s approach and adapt their method of projecting dependency structures to the LFG framework. Two main characteristics of LFG make it especially suitable for this cross-lingual projection method: (i) Since LFG is a lexicalized theory, projection of annotations assigned to particular words can be sufficently guided by word alignment. (ii) F-structures constitute an abstract level of analysis that is largely invariant across languages, and thus perfectly suited for projection between languages with varying word order.

LFG f-structures encode grammatical functions holding between PREDicates and their arguments or modifiers, represented as partial f-structures. Next to PREDs, partial f-structures encode morpho-syntactic information, such as CASE, TENSE, PERSon and NUMber. The close correspondence between grammatical function information in LFG and syntactic dependencies in general is most easily seen by viewing grammatical functions holding between partial f-structures as relating two lexical items (PREDs) that head the corresponding partial f-structures.[4]

Cross-lingual projection of f-structures is defined as follows: Projection is grounded in automatically induced word alignment links $al(te_i, tf_i)$ $(al \in AL : E \times F)$ between English and Polish surface words (terminals) $te_i \in E, tf_i \in F$. Based on the LFG-parsed English corpus, we identify the corresponding lexical items (PREDs) e in the English f-structures. The set GF consists of grammatical functions SUBJ, OBJ, OBL, ADJ, etc. that hold between pairs of English PREDicates $gf(e_i, e_j)$. During projection, grammatical functions gf encoded in the source f-structure fs_e are transferred to the target sentence via word alignment links, according to the following definition, which is similar to the one in Hwa et al. [7]:

> The grammatical function $gf(e_i, e_j)$ holding between PREDS e_i and e_j in the source f-structure fs_e projects to the target f-structure fs_f as $gf(f_i, f_j)$ if and only if the source terminals te_i and te_j that project to the PRED values e_i, e_j included in fs_e are aligned with the target terminals tf_i and tf_j of f_i and f_j, respectively.[5]

[3]However, since Polish has an extended case system distinguishing seven cases, studying the interaction of grammatical functions and case marking will be important for further improvements.

[4]In general, dependency structures are assumed to be trees, whereas LFG f-structures are graphs. This difference has no effect on the present projection approach.

The definition states that if two English words are related by a grammatical function, the same grammatical function will relate their word counterparts in Polish. Similar to Hwa et al. [7], we define specific projection constraints for different types of alignment links:[6]

one-to-one: a grammatical function $gf(e_i, e_j)$ relates source words e_i and e_j that are aligned with exactly one target word $al(e_i, f_i)$ and $al(e_j, f_j)$, respectively.

one-to-many: a grammatical function $gf(e_i, e_j)$ relates source words e_i and e_j that are aligned with a single target word: $al(e_i, f_i)$ and $al(e_j, f_i)$. The gf is projected to the partial target f-structure fs_i headed by f_i: $gf(f_i, f_j)$, where f_j is defined as [PRED 'pro'] for an incorporated SUBJ pronoun (pro-drop) or as [PRED 'null'] for other grammatical functions (e.g. null specifiers SPEC).

unaligned e: a grammatical function $gf(e_i, e_j)$ relates source words e_i and e_j, where e_i is aligned with the target word f_i $al(e_i, f_i)$, the other with *none* $al(e_j$, *none*). The gf is projected to the partial f-structure fs_i headed by f_i: $gf(f_i, f_j)$. f_j is defined as [PRED 'pro'] for an incorporated SUBJ pronoun or as [PRED 'null'] for other grammatical functions (e.g. null specifiers SPEC).

However, the *Direct Correspondence Assumption* that underlies the annotation projection approach is an idealisation. Indeed, the projected grammatical functions may be incorrect, due to

(i) errors in the source annotations obtained from automatic LFG parsing;

(ii) poor accuracy of automatic word alignment;

(iii) true mismatches of functional structure between English and Polish.

These error sources radically impair the quality of the projected grammatical functions. However, these shortcomings can be overcome by applying correction rules similar to Hwa et al. [7] that locally transform the induced Polish f-structures. We have defined two post-projection correction rules that are motivated by general linguistic properties of the Polish language:[7]

Rule 1: The PRED value of the SPEC_DET function for the article 'the' or 'a/an' in English is replaced by 'null' in the Polish partial f-structure (cf. Figure 1).

Rule 2: The grammatical function borne by an *of*-prepositional phrase in English is realized by a genitive noun phrase in Polish (cf. Figure 2).

[5]This definition presupposes lemmatisation on the target side, to provide appropriate values f_i for the target f-structure's PRED features. In practice, given that we don't use any tagger or morphological analyzer to preprocess the Polish corpus, the target f-structure PRED values are instantiated with the aligned Polish surface words (together with the English lemma, for better readability, see below Figures 1-3). In the following, we will use f_i, f_j instead of tf_i, tf_j, to keep the definitions simpler.

[6]Hwa et al. 2005 distinguish 5 alignment type scenarios: one-to-one, unaligned (English), one-to-many, many-to-one, many-to-many. In our unidirectional alignment experiment (cf. Section 4), we only found the 3 alignment types defined above.

[7]Further correction rules may be formulated by taking into account morpho-syntactic information concerning case, number, tense etc. Currently, we do not consider morpho-syntactic features.

$$\begin{bmatrix} \text{PRED} & \text{'cecha (feature)'} \\ \text{SPEC\_DET} & \begin{bmatrix} \text{PRED} & \text{'lex (the)'} \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} \text{PRED} & \text{'cecha (feature)'} \\ \text{SPEC\_DET} & \begin{bmatrix} \text{PRED} & \text{'null'} \end{bmatrix} \end{bmatrix}$$

Figure 1: Example of application of Rule 1: an erroneously induced Polish article equivalent to English 'the' in SPEC_DET function is replaced by a 'null' specifier.

$$\begin{bmatrix} \text{PRED} & \text{'decyzja (decision)'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{'komitetu (of)'} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'komitetu (committee)'} \end{bmatrix} \end{bmatrix} \right\} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{PRED} & \text{'decyzja (decision)'} \\ \text{ADJUNCT} & \left\{ \begin{bmatrix} \text{PRED} & \text{'komitetu (committee)'} \end{bmatrix} \right\} \end{bmatrix}$$

Figure 2: Example of application of Rule 2: the prepositional projection level of an *of*-PP in English is reduced to yield an NP adjunct in Polish.

We currently focus on the projection of grammatical functions, without considering morpho-syntactic features. The induced Polish f-structures are therefore *preds-only f-structures*. The following f-structure of the Polish sentence *Niniejsza dyrektywa skierowana jest do Państw Członkowskich* is automatically induced based on the f-structure of its English equivalent 'This directive is addressed to the Member States.'

$$\begin{bmatrix} \text{PRED} & \text{'skierowana\_jest (address)'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'dyrektywa (directive)'} \\ \text{SPEC\_DET} & \begin{bmatrix} \text{PRED} & \text{'niniejsza (this)'} \end{bmatrix} \end{bmatrix} \\ \text{OBL} & \begin{bmatrix} \text{PRED} & \text{'do (to)'} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'państw (state)'} \\ \text{MOD} & \left\{ \begin{bmatrix} \text{PRED} & \text{'członkowskich (member)'} \end{bmatrix} \right\} \\ \text{SPEC\_DET} & \begin{bmatrix} \text{PRED} & \text{'null'} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3: The f-structure of the Polish sentence *Niniejsza dyrektywa skierowana jest do Państw Członkowskich.* ('This directive is addressed to the Member States.')

Based on the projection of f-structure information as stated above, and enhanced with post-correction rules as seen in Figures 1 and 2, we obtain an LFG f-structure bank for Polish that may be used to train a dependency parser, or a full-fledged LFG c- and f-structure bank and LFG grammar, by inducing f- to c-structure mappings for Polish, along the lines of Klein [8] for English[8] and inducing a probabilistic treebank-based LFG grammar, using the method of Cahill et al. [4].

[8]The f-structures used in Klein [8] contain morpho-syntactic features based on PoS information. In a similar way, our proto-f-structures induced for Polish can be enriched with morphological information using a PoS-tagger for Polish (on the target language side).

4 Data, Evaluation and Results

4.1 Data and preprocessing

Our projection experiment is conducted on the JRC-Acquis Multilingual Parallel Corpus [13], a large collection of European Union legislative texts that – unlike Europarl – includes texts in Polish. From the full English-Polish section of JRC-Acquis (1,26 mil. sentence links) we selected a subcorpus aligned on the sentence level consisting of 257,144 sentence pairs. The average sentence length ranges from 4 to 30 tokens. In the preprocessing phase the parallel texts are word-aligned, the English side of the bi-text is parsed into LFG f-structures, and we apply some filtering methods. We briefly describe these phases, in turn.

Word alignment is performed with the SMT system MOSES [9], based on statistics captured from the entire corpus. To a certain degree English is an isolating language that makes use of function and non-content words. Polish, by contrast, is a highly inflecting language and needs in general fewer or as many words as English to express the same content. In order to decide whether alignment of one Polish word with one or more English words conforms to general translational mappings, we use unidirectional Polish-English word alignment as a basis for projection.

LFG parsing The English side of the parallel corpus is parsed with the hand-crafted wide-coverage English LFG grammar, which is enhanced with a statistical disambiguation component selecting the most probable analysis.

Filtering We filter all duplicates and omit sentences that contain inconsistencies of tokenization between MOSES and XLE.

4.2 Evaluation

We evaluate the quality of the automatically induced f-structures against a gold standard consisting of f-structures of 50 Polish sentences randomly selected from the entire corpus. The gold f-structures chosen from the preprocessed data (11.98 tokens/sent. for English, 9.76 tokens/sent. for Polish) have been manually corrected. For this purpose, the f-structures were first transformed to the SALSA/TIGER XML format required by the SALTO annotation tool (Burchardt et al. [2]), which enables efficient modification by adding, deleting or correcting grammatical functions. We calculate precision, recall and f-score for various projection scenarios (cf. Table 1):

+/- correction: for exact match of projected grammatical functions, distinguishing direct projection (**direct**) and projection with post-modification (**+corr**) using the two correction rules mentioned above;

+/- automatic: for the projected grammatical functions taking into account automatically derived (noisy) word alignment (**automatic**) in contrast to hand-corrected (optimal) word alignment (**manual**), to establish an upper bound.

experiment	languages	sentences (corpus)	alignment	projection	LP /ULP	LR /ULR	F-score /UF-score
Current Experiment	en-pl	50 (JRC-Acquis)	automatic	direct	49/50	51/52	50/51
				+corr	64/64	63/63	63.5/63.5
			manual	direct	61/62	63/63	62/62.5
				+corr	85/85	81/82	83/83.5
Hwa et al.	en-sp	100 (UN/FBIS /Bible)	automatic	direct			/33.9
				+corr			/65.7
			manual	direct			/36.8
				+corr			/70.3
Ozdowska	en-pl	50 (AC)	automatic	direct	67/82		
Bouma et al.	ge-de	222 (Europarl)	automatic	direct	52.2	52.9	52.6
	en-du				54.3	48.8	51.4
	(ge,en)-du				74.6	34.1	46.8

Table 1: LP/ULP: precision for labeled/unlabeled grammatical functions; LR/ULR: recall for labeled/unlabeled grammatical functions.

Results. As expected, direct projection of grammatical functions is noisy (49.98% f-score). Application of language-specific transformation rules considerably improves the accuracy of the projected grammatical functions (63.5% f-score). The *quality of word alignment* is a crucial factor for projection quality: projection based on corrected word alignments enhances the quality of the induced f-structures by 12 percentage points (pp) f-score for direct projection and by 19.45 pp f-score for projection with transformation rules. In line with Hwa et al. [7], these results clearly indicate that direct projection of grammatical functions is significantly outperformed by projection using *post-projection transformation rules*, both for automatic word alignment (13.52 pp f-score improvement), and for manually corrected word alignments, the latter constituting an upper bound of 20.97 pp f-score improvement. The upper bound (projection based on perfect word alignment) in conjunction with two correction rules yields an accuracy of 82.95% f-score.

4.3 Error Analysis

According to our error analysis, most errors are due to word alignment mistakes, especially wrong alignment of a post-modifier of a Polish noun with an English head noun. Further, errors in the automatically parsed English f-structures have a big impact, as grammatical functions are projected to Polish without quality tests. A final source of errors are translational divergences. They are caused by language-specific conventions or structural constraints on how to express the same content in different languages, or simply by some “translational freedom” taken by translators. Translational divergences may radically change the syntactic structure of a translated sentence as compared to its source. Erroneous f-structures caused by word alignment errors, mistakes in the source analysis or translational divergences may be filtered on the target side, if morphological or PoS information is provided.

4.4 Comparison to Related Work

A full comparison to related approaches is difficult, due to the different languages and corpora involved. Keeping this in mind, compared to the results of unlabeled dependency projection for Spanish in Hwa et al. [7], we obtain comparable f-scores (63.5 vs. 65.7) for automatic projection with post-correction. For direct projection, we outperform Hwa et al.'s results by 17 pp. Compared to Ozdowska [10], precision of our direct projection is lower by 18/32 pp. Ozdowska [10] relies on one-to-one alignment links (intersection) only, which increases precision but decreases recall. As Ozdowska [10] does not report recall figures, we cannot compare the results. Bouma et al. [1] represents an LFG-based approach, like ours. However, their work is restricted to verb arguments while we perform full f-structure induction. We observe that combining argument information from two languages (English and German) enhances precision but degrades recall. In contrast, we obtain balanced precision and recall values. Regarding f-score, our projection of grammatical functions outperforms the projection by Bouma et al. [1] by 16.7 pp.

5 Conclusions and Future Work

In summary, we presented a cross-lingual projection approach for creating an LFG f-structure bank for Polish. Our results are competitive as compared to related prior work on different languages and corpora. Similar to Hwa et al.'s work, the application of post-correction rules significantly improves the quality of the induced f-structures obtained by direct projection. It is worth mentioning that we obtain high, balanced precision and recall values. Based on the gold standard word alignment, we identified an upper bound of 83% f-score to build a Polish f-structure bank. We find that word alignment is a crucial factor affecting the accuracy of the projected grammatical functions, next to principled linguistic differences. In fact, there is a stark contrast between Hwa et al.'s delta to achieve the gold standard level (70.3% vs. 65.7%) and ours (83.5% vs. 63.5%). This indicates that word alignment constitutes a harder problem for alignment of English with Polish as compared to Spanish. Linguistic differences may be addressed by extending the set of post-correction rules. Word alignment may be improved by advances in the state of the art, e.g. using lemmatised and PoS-tagged corpora for word alignment.

In future work, we will explore inclusion of problematic data (inconsistent tokenization), improvement of word alignment, and use of morpho-syntactic information to further enhance the projection quality. The resulting Polish f-structure bank may be efficiently used to train a dependency parser. Training on Hwa et al.'s Spanish data (with 65.7% f-score) yielded a parsing performance of 72.1% f-score [7]. Since our projection model approaches comparable f-score (63.5%), we expect that a dependency parser for Polish will achieve comparable performance, with potential increase by further improvement of the base projection quality. Finally, we will explore induction of a full-fledged LFG grammar for Polish, by adding a module that learns f-to-c-structure mappings for Polish, along the lines of Klein [8].

References

[1] G. Bouma, J. Kuhn, B. Schrader, and K. Spreyer. Parallel LFG Grammars on Parallel Corpora: A base for practical triangulation. In M. Butt and T.-H. King, editors, *Proceedings of the LFG 2008 Conference*, pages 169–189, Sydney, 2008.

[2] A. Burchardt, K. Erk, A. Frank, A. Kowalski, S. Padó, and M. Pinkal. SALTO - A Versatile Multi-Level Annotation Tool. In *Proceedings of LREC 2006*, pages 517–520, 2006.

[3] M. Butt, H. Dyvik, T.H. King, H. Masuichi, and Rohrer Ch. The Parallel Grammar Project. In *Proceedings of the COLING 2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, Taiwan, 2002.

[4] A. Cahill, M. Forst, M. Burke, M. McCarthy, R. O'Donovan, C. Rohrer, J. van Genabith, and A. Way. Treebank-Based Acquisition of Multilingual Unification Grammar Resources. *Journal of Research on Language and Computation*, 3(2):247–279, 2005.

[5] M. Darlymple and R.M. Kaplan. Feature Indeterminacy and Feature Resolution. *Language*, 76(4):759–798, 2000.

[6] M. Diab and P. Resnik. An Unsupervised Method for Word Sense Tagging using Parallel Corpora. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, ACL 2002*, pages 255–262, Philadelphia, 2002.

[7] R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping Parsers via Syntactic Projection across Parallel Texts. *Natural Language Engineering*, 11(3):311–325, 2005.

[8] A. Klein. Von Dependenzstrukturen zu Konstituentenstrukturen: Automatische Generierung von Penn-Treebank-Bäumen aus LFG-F-Strukturen. Master's thesis, Universität Heidelberg, 2009.

[9] P. Koehn, M. Federico, W. Shen, N. Bertoldi, O. Bojar, Ch. Callison-Burch, B. Cowan, Ch. Dyer, H. Hoang, R. Zens, A. Constantin, Ch.C. Moran, and E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL 2007*, pages 177–180, Prague, 2007.

[10] S. Ozdowska. Projecting POS Tags and Syntactic Dependencies from English and French to Polish in Aligned Corpora. In *Proceedings of the EACL 2006 Workshop on Cross-Language Knowledge Induction*, pages 53–60, Trento, 2006.

[11] S. Padó and M. Lapata. Cross-linguistic Projection of Role-semantic Information. In *Proceedings of HLT/EMNLP 2005*, pages 859–866, Vancouver, 2005.

[12] K. Spreyer and A. Frank. Projection-based Acquisition of a Temporal Labeller. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008)*, pages 489–496, Hyderabad, India.

[13] R. Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, Tufis D., and D. Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of LREC 2006*, pages 2142–2147, Genoa, Italy, 2006.

[14] D. Yarowsky and G. Ngai. Inducing Multilingual POS Taggers and NP Bracketers via Robust Projection across Aligned Corpora. In *Proceedings of NAACL 2001*, pages 200–207, 2001.