The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018

EDITORIAL BOARD

Published twice a year by Charles University (Prague, Czech Republic)

Editorial office and subscription inquiries:
ÚFAL MFF UK, Malostranské náměstí 25, 118 00, Prague 1, Czech Republic
E-mail: pbml@ufal.mff.cuni.cz

ISSN 0032-6585

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018

CONTENTS

Articles

Using Tectogrammatical Annotation for Studying Actors and Actions in Sallust's Bellum Catilinae 5
Berta González Saavedra, Marco Passarotti

Enriching VALLEX with Light Verbs: From Theory to Data and Back Again 29
Václava Kettnerová, Markéta Lopatková, Eduard Bejček, Petra Barančíková

PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing 57
Lauriane Aufrant, Guillaume Wisniewski

An Easily Extensible HMM Word Aligner 87
Jetic Gū, Anahita Mansouri Bigvand, Anoop Sarkar

A Probabilistic Approach to Error Detection&Correction for Tree-Mapping Grammars 97
Tim vor der Brück

NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning 113
Álvaro Peris, Francisco Casacuberta

Open Source Toolkit for Speech to Text Translation 125
Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller, Ngoc-Quan Pham, Sebastian Stüker, Alex Waibel

Instructions for Authors 136

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 5–28

Using Tectogrammatical Annotation for Studying Actors and Actions in Sallust's Bellum Catilinae

Berta González Saavedra,[a] Marco Passarotti[b]

[a] Dep. de Filología Clásica, Universidad Autónoma de Madrid, Spain
[b] CIRCSE Research Centre. Università Cattolica del Sacro Cuore, Milan, Italy

Abstract

In the context of the *Index Thomisticus* Treebank project, we have enhanced the full text of *Bellum Catilinae* by Sallust with semantic annotation. The annotation style resembles the one used for the so called "tectogrammatical" layer of the Prague Dependency Treebank. By exploiting the results of semantic role labeling, ellipsis resolution and coreference analysis, this paper presents a network-based study of the main Actors and Actions (and their relations) in *Bellum Catilinae*.

1. Introduction

Since the second half of the nineties, the research area dealing with enhancing linguistic data with syntactic annotation ("treebanking") has faced a turn from constituency-based to dependency-based annotation schemata. The result is the current availability of several dependency treebanks for quite a number of languages. Most of these are now part of Universal Dependencies (`http://universaldependencies.org/`), an ever growing collection of dependency treebanks for several different languages following a cross-linguistically consistent annotation schema, which is in the process of becoming the standard de facto in the field.

The large majority of the currently available treebanks includes data taken from contemporary books, magazines, journals and, mostly, newspapers. Such data are

This paper is an extended version of the work presented by Passarotti and González Saavedra (2017) at the Sixteenth Edition of the International Workshop on Treebanks and Linguistic Theories (TLT-16), 23-24 January 2017, Prague, Czech Republic.

 Corresponding author: marco.passarotti@unicatt.it
Cite as: Berta González Saavedra, Marco Passarotti. Using Tectogrammatical Annotation for Studying Actors and Actions in Sallust's Bellum Catilinae. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 5–28.
doi: 10.2478/pralin-2018-0005.

used for different purposes in both theoretical and computational linguistics, the most widespread being supporting and evaluating theoretical assumptions with empirical evidence and providing data for various tasks in stochastic Natural Language Processing (NLP), like inducing grammars and training/testing tools.

Throughout the last decade, a small but constantly growing bunch of dependency treebanks for ancient languages was built. In this respect, the main treebanks now available are those for Latin and Ancient Greek, with The Ancient Greek and Latin Dependency Treebank (AGLDT) (Bamman and Crane, 2011), the *Index Thomisticus* Treebank (IT-TB) (Passarotti, 2011) and the PROIEL corpus (Haug and Jøhndal, 2008). Moreover, dependency treebanks are available also for other ancient languages, like Gothic and Old Church Slavonic (part of PROIEL), and Hittite (Inglese, 2015). Such linguistic resources for ancient languages support studies in historical linguistics together with a number of treebanks that include texts representing different diachronic phases of modern languages. Examples are the York-Toronto-Helsinki corpus (Taylor, 2007) and the Penn Corpora of Historical English (Taylor and Kroch, 1994; Kroch et al., 2004), the MCVF corpus for French (Martineau, 2008), the Tromsø Old Russian and OCS Treebank (Eckhoff and Berdicevskis, 2015) and RRuDi for Russian (Meyer, 2011), and the Mercurius Treebank for Early High New German (Demske, 2007).

Unlike those for modern languages, treebanks for ancient languages tend to include literary, historical, philosophical and/or documentary texts. This makes the very use of such resources peculiar. Indeed, instead of exploiting data to draw (cross-)linguistic generalizations, the users of such treebanks are more interested in the linguistic features of the texts themselves available in the corpus. For instance, there is more interest and scientific motivation in exploiting the treebanked texts of Sophocles to study their specific syntactic characteristics than in using the evidence provided by such texts as sufficiently representative of Ancient Greek, which they are not.

Not only the use of data is different, but also users are. Indeed, it is quite uncommon that scholars from literature, philosophy or history make use of linguistic resources like treebanks for modern languages in their research work. Instead, they represent some of the typical users of treebanks for ancient languages as well as of diachronic treebanks. Such resources become even more useful for this kind of users from the Humanities when they are enhanced also with a semantic layer of annotation, on top of the syntactic one. This is due to the large interest of such scholars in semantic interpretation of texts through syntax.

In this area, the *Index Thomisticus* Treebank project has recently enhanced a selection of texts taken from the IT-TB and the AGLDT with semantic annotation. This paper describes the dependency-based annotation style applied on these data and presents a use case of exploitation of them for literary analysis purposes. In particular, by using the results of semantic role labeling, coreference analysis and ellipsis resolution applied on the source data, the analysis focuses on the main Actors and Actions in Sallust's *Bellum Catilinae*.

Written probably between 43 and 40 BCE, *Bellum Catilinae* tells the story of the so called Second Catilinarian Conspiracy (63 BCE), a plot, devised by Catiline and a group of aristocrats and veterans, to overthrow the Roman Republic.[1] One of the masterpieces of the Latin literature, *Bellum Catilinae* has been object of several and exhaustive studies, especially by historians of the Roman republican period and scholars in Latin Literature and Linguistics. From a historical perspective, particular attention has been paid to the intention of Sallust when writing his book (Conley, 1981) as well as to the amount of historical incongruences in his text (Syme, 1964). In addition, multiple contributions focus on various aspects of the language of Sallust (Batstone, 2010; Schröder, 2015; Tannenbaum, 2005).

The figure of Catiline has always fascinated the general public, particularly because of the complexities of his character as organizer of the conspiracy. Thus, a significant number of works deal with Sallust's depiction of Catiline, most of the times comparing it to the one provided by Cicero in his *In Catilinam*, which was no doubt the most important source for Sallust while writing *Bellum Catilinae*.[2]

Precisely because of such kind of portrayals, throughout the centuries the image of Catiline was deformed to the point that modern scholars often considered him an evil character (Earl, 1958) as well as the personification of ambition and greed (McConaghy, 1974). Contrary to these approaches, Ann Thomas Wilkins (1994) proposed in the nineties to read Sallust's treatment of the character of Catiline in a more complex way, according to which the author would be using the conspiracy of Catiline with the clear intention to show the decadence of Rome. In the view of Wilkins, the purposes of Sallust lay on the structure of the work, as she creates an antithesis between the first part of the account -where Catiline's conspiracy is presented from the perspective of Roman oligarchy– and the second part –where the distinction between the revolutionaries and the members of the establishment is already blurred. To this aim, Wilkins focuses on the distribution of the book, considering narrative periods, discourses, moral digressions and, most of all, how the descriptive words used for and by the main characters become common for both sides.

Moving from such a linguistic-based approach, we believe that analyzing the main Actors and Actions in *Bellum Catilinae* through the (deep) semantic annotation of the entire text of Sallust can provide a strong empirical support helping historians and Literature scholars to shed some further light on Sallust's portrayal of Catiline.

---

[1] The text of *Bellum Catilinae* available from the AGLDT is the one edited by Ahlberg (1919). It includes 10,936 words and 701 sentences. In this paper, English translations of *Bellum Catilinae* are taken from Ramsey (2014).

[2] On this question, see Broughton (1936) and Waters (1970). An interesting perspective is provided by Syme (1964; page 73), who defends that Cicero is not the only author Sallust used for the compilation of *Bellum Catilinae*.

2. Data

In the context of the *Index Thomisticus* Treebank project hosted at the CIRCSE research centre of the Università Cattolica del Sacro Cuore in Milan, Italy (`http://itreebank.marginalia.it/`), we have added a new layer of semantic annotation on top of a selection of syntactically annotated data taken from the IT-TB and the Latin portion of the AGLDT (González Saavedra and Passarotti, 2014).

In particular, around 2,000 sentences (approx. 27,000 words) were annotated out of *Summa contra Gentiles* of Thomas Aquinas (IT-TB). The entire *Bellum Catilinae* of Sallust (BC) and small excerpts of 100 sentences each from texts of Caesar and Cicero were annotated from the AGLDT.

2.1. Annotation Style

The style of the semantic layer of annotation used in the IT-TB project is based on Functional Generative Description (FGD) (Sgall et al., 1986), a dependency-based theoretical framework developed in Prague and intensively applied and tested while building the Prague Dependency Treebank of Czech (PDT) (Hajič et al., 2000).

The PDT is a dependency-based treebank with a three-layer structure. The (so ordered) layers are a "morphological layer" (morphological tagging and lemmatization), an "analytical" layer (annotation of surface syntax) and a "tectogrammatical" layer (annotation of underlying syntax). Both the analytical and the tectogrammatical layers describe the sentence structure with dependency tree-graphs, respectively named analytical tree structures (ATSs) and tectogrammatical tree structures (TGTSs).

In ATSs every word and punctuation mark of the sentence is represented by a node of a rooted dependency tree. The edges of the tree correspond to dependency relations that are labelled with (surface) syntactic functions called "analytical functions" (like Subject, Object etc.).

TGTSs describe the underlying structure of the sentence, conceived as the semantically relevant counterpart of the grammatical means of expression (described by ATSs). The nodes of TGTSs include autosemantic words only (represented by "tectogrammatical lemmas": "t-lemmas"), while function words and punctuation marks collapse into the nodes for autosemantic words. Semantic role labeling is performed by assigning to nodes semantic role tags called "functors". These are divided into two classes according to valency: (a) arguments, called "inner participants", i.e. obligatory complementations of verbs, nouns, adjectives and adverbs: Actor,[3] Patient, Ad-

[3]The definition of Actor in the PDT is semantically quite underspecified, as it refers to "the human or non-human originator of the event, the bearer of the event or a quality/property, the experiencer or possessor" (Mikulová et al., 2006; page 461).

dressee, Effect and Origin; (b) adjuncts, called "free modifications": different kinds of adverbials, like Place, Time, Manner etc.[4]

Also coreference analysis and ellipsis resolution are performed at the tectogrammatical layer and are represented in TGTSs through arrows (coreference) and newly added nodes (ellipsis). In particular, there are two kinds of coreference: (a) "grammatical coreference", in which it is possible to pinpoint the coreferred expression on the basis of grammatical rules (mostly with relative pronouns) and (b) "textual coreference", realized not only by grammatical means, but also via context (mostly with personal pronouns).

2.2. From Analytical to Tectogrammatical Layer

2.2.1. Converting from ATSs to TGTSs in the *Index Thomisticus* Treebank Project

The workflow for tectogrammatical annotation in the IT-TB project is based on TGTSs automatically converted from ATSs.[5] The TGTSs that result from the conversion are then checked and refined manually by two annotators. The conversion is performed by adapting to Latin a number of ATS-to-TGTS conversion modules provided by the NLP framework *Treex* (Žabokrtský, 2011).[6]

Relying on ATSs, the basic functions of these modules are the following:

a. to collapse ATSs nodes of function words and punctuation marks, as they no longer receive a node for themselves in TGTSs, but collapse into the nodes for autosemantic words;
b. to assign "grammatemes", i.e. semantic counterparts of morphological categories (for instance, pluralia tantum are tagged with the number grammateme "singular");
c. to resolve grammatical coreferences;
d. to assign semantic roles.

Tasks (a) and (b) are quite simple and the application of the modules that are responsible for them results in good accuracy on average. Collapsing nodes for function

[4]The organization of functors into inner participants and free modifications is further exploited by linking textual tectogrammatical annotation with fundamental lexical information provided by a valency lexicon that features the valency frame(s) for all those verbs, nouns, adjectives and adverbs capable of valency that occur in the treebank. The valency lexicon of Latin, called *Latin Vallex* (Passarotti et al., 2016), was built in corpus-driven fashion, by adding to the lexicon all the valency-capable words that annotators progressively got through. A similar approach to build a valency lexicon based on treebank annotation is that of *PDT-Vallex* for Czech (Urešová, 2009).

[5]The guidelines for analytical annotation of the IT-TB (as well as of the Latin portion of the AGLDT) are those of Bamman et al. (2007). The guidelines for tectogrammatical annotation are those of the PDT (Mikulová, 2006), with a few modifications for representing Latin-specific constructions (`http://itreebank.marginalia.it/doc/Guidelines_tectogrammatical_Latin.pdf`).

[6]See González Saavedra and Passarotti (2014) for details on ATS-to-TGTS conversion in the IT-TB and, especially, for an evaluation of the accuracy of the conversion process.

words and punctuations relies on the structure of the ATSs given in input. In this respect, Latin does not feature any specific property requiring for modifications of the ATS-to-TGTS conversion procedure available in *Treex* and already applied to other languages. Assigning grammates is a task strictly related with the lexical properties of the nodes in TGTSs. Thus, we are in the process of populating the modules that assign grammatemes with lists of words (lemmas) that are regularly assigned the same grammatemes.

The automatic processing of task (c) results from the application of a number of modules aimed to resolve only the grammatical coreference that shows the simplest possible construction occurring in ATSs, i.e. the one featuring an occurrence of a relative pronoun directly depending on the main predicate of the relative clause. However, this construction is highly frequent for relative clauses. For instance, among the 326 occurrences of the relative pronoun *qui* in the portion of the IT-TB featuring tectogrammatical annotation, 176 present this construction and are correctly assigned their grammatical coreference by the conversion modules. The remaining 150 occurrences either lack grammatical coreference or do occur in more complex constructions.

In order to assign semantic roles automatically (task (d)), we rely both on analytical functions and on lexical properties of the ATSs nodes. For instance, all the nodes with analytical function Sb (Subject) that depend on an active verb are assigned functor ACT (Actor), and all the main predicates of subordinate clauses introduced by the conjunction *si* 'if' are assigned functor COND (Condition).

2.2.2. Examples of ATSs and TGTSs from *Bellum Catilinae*

In this section we report a number of examples of ATSs and TGTSs from BC.

Figure 1 shows the ATS for the sentence "Sed [but] maxume [most of all] adulescentium [of the young] familiaritatem [intimacy] adpetebat [sought]" (BC 14.5) ("But most of all [Catiline] sought the intimacy of young men").

The ATS in Figure 1 features as many nodes as the words of the sentence (5) plus the root node, which reports the ID of the sentence in the Latin portion of the AGLDT ("a-" here means "analytical") and it is assigned by default the analytical function AuxS (Sen-

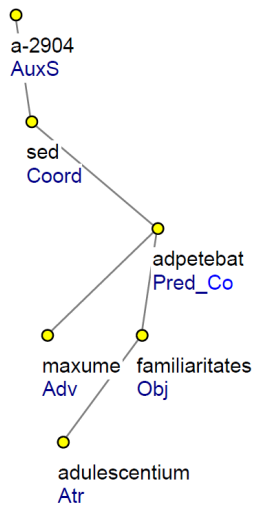


Figure 1. ATS of the sentence "Sed maxume adulescentium familiaritatem adpetebat" (BC 14.5).

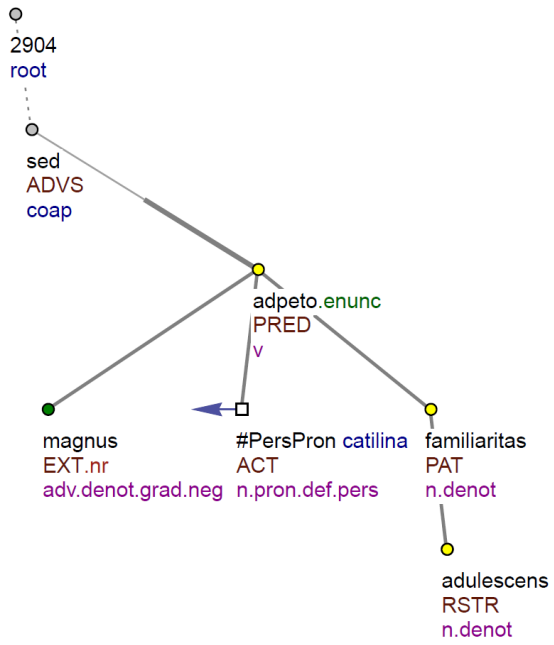


Figure 2. TGTS of the sentence "Sed maxume adulescentium familiaritatem adpetebat" (BC 14.5).

tence). Nodes are arranged from left to right according to the order of the words in the sentence. Each node is assigned an analytical function.[7]

The TGTS shown in Figure 2 features all the nodes of the corresponding ATS plus one. This newly added square node results from both ellipsis resolution and coreference analysis.

As for the former, the square node fills the position for a missing argument of the verb *adpeto*. Here "missing" means that the argument is not explicitly represented by either a lexical item or a phrase in the text. In this sentence, the verb *adpeto* is considered a word with two arguments, which are represented respectively by an Actor (ACT: missing) and a Patient (PAT: *familiaritas*).

As for the latter, the newly added node for the missing Actor is assigned t-lemma #PersPron,[8] which means that the node represents the missing occurrence of a per-

[7]Coord: coordination. Pred_Co: coordinated main predicate. Adv: adverbial modifier (adjunct). Obj: direct or indirect object (argument). Atr: attributive.

[8]#PersPron is the t-lemma assigned to nodes representing possessive and personal pronouns (including reflexives). Sets of different morphological lemmas can be grouped under the same t-lemma in TGTSs. This

sonal pronoun (like *is* 'he'), which is permitted by the pro-drop nature of Latin. The node is linked via a textual coreference to the last previous occurrence of the lemma *catilina*, which represents its denotation.

In Figure 2, nodes are arranged from left to right reflecting information structure according to Topic-Focus Articulation, moving from Topic (left) to Focus (right).[9] Each node is assigned a functor and a so called "semantic part of speech". The occurrence of the lemma *magnus* (form *maxume*) represents an EXT (Extent), i.e. an adjunct that expresses manner by specifying extent or intensity of the event or a circumstance. The semantic part of speech for this occurrence is that for gradable adverbs that can be negated. *Familiaritas* is a denominating semantic noun (n.denot) further specified by another noun acting as a restrictor of its head in the TGTS (functor: RSTR). Finally, *sed* is a an adversative (ADVS) coordinating connective (coap).

The main predicate of the sentence is assigned the so called "sentential modality", which consists in speech act annotation. In the TGTS shown in Figure 2, the sentence is an "enunciation" (enunc).

Figure 3 shows the ATS for the sentence "Sed [but] iuventutem [the young], quam [whom], ut [as] supra [above] diximus [we said], illexerat [he had ensnared], multis [many] modis [ways] mala [bad] facinora [crimes] edocebat [he taught]" (BC 16.1) ("The young men whom he had ensnared, as I have mentioned above, were instructed by him in wicked deeds of many forms").

The only analytical functions in Figure 3 that do not occur also in Figure 1 are AuxX (assigned to punctuation marks) and AuxC (for subordinating conjunctions). Figure 4 shows the corresponding TGTS.

In this sentence, *catilina* is Actor of two verbs: *illicio* and *edoceo*. In both cases, pronoun dropping and ellipsis resolution is performed. The Actor of

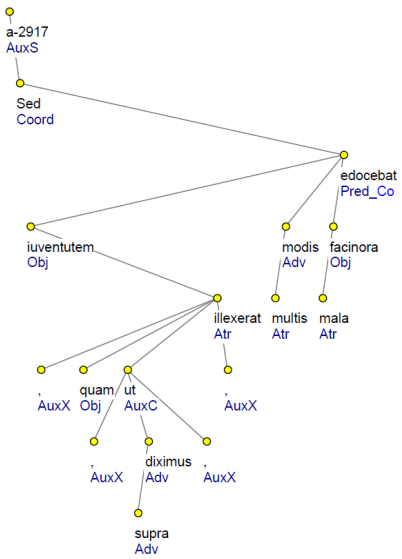


Figure 3. ATS of the sentence "Sed iuventutem, quam, ut supra diximus, illexerat, multis modis mala facinora edocebat" (BC 16.1).

is the case, for instance, of morphological lemmas *aliquis* 'someone', *quis* 'who?' 'which?', *quisquis* 'whoever' and *unusquisque* 'each', which are all assigned t-lemma *quis*.

[9]For details about Topic-Focus Articulation, see Mikulová et al. (2006; pages 1118-1188)

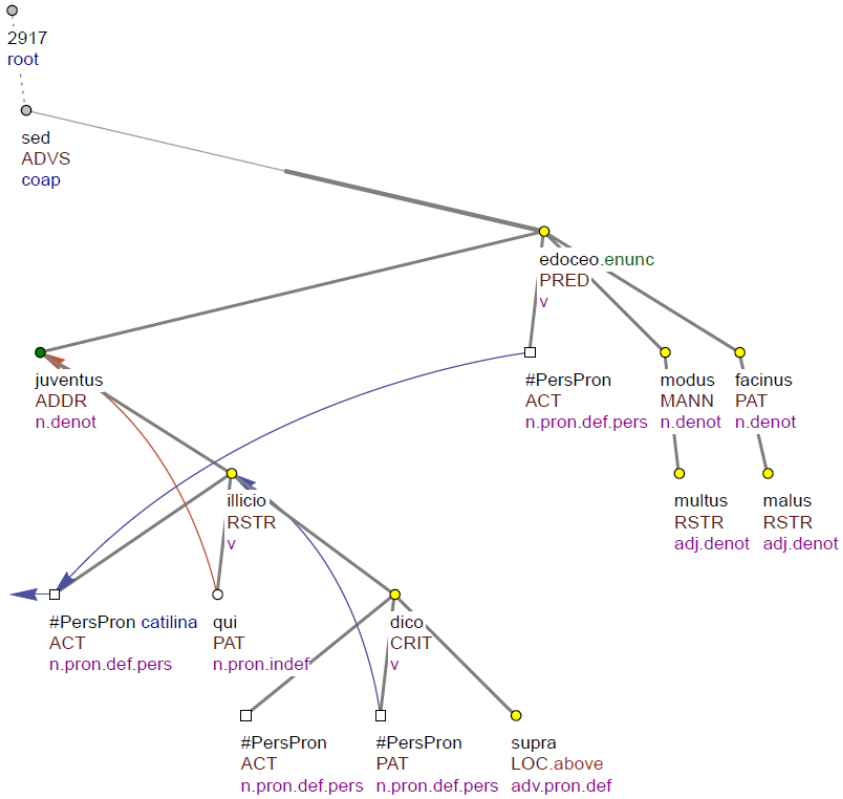


Figure 4. TGTS of the sentence "Sed iuventutem, quam, ut supra diximus, illexerat, multis modis mala facinora edocebat" (BC 16.1).

edoceo is linked via a textual coreference to that of *illicio*, which is in turn textually coreferred to the previous occurrence of *catilina*. Equally, the newly added node standing for the Patient of the verb *dico* is linked to *illicio*, because what "we have said above" is that "he had ensnared the young men". Figure 4 shows also a grammatical coreference holding between the relative pronoun *qui* (Patient of *illicio*) and the noun *juventus* (Addresse of *edoceo*). As for the functors, LOC is assigned to Locatives answering the question "where?" and it is here further specified by the subfunctor "above" (*supra*). MANN is a functor for such an adjunct that expresses manner (*modis*). Finally, it is worth noting that the TGTS of Figure 4 does not include the node for the function word *ut*, which collapses into that for the autosemantic word *dico*.

Figure 5 shows the ATS for the sentence "cum [with] eo [him] se [himself] consulem [consul] initium [beginning] agundi [of acting] facturum [would have made]" (BC 21.4) ("[Catiline promised that] as consul with him, he would launch his undertaking"), which presents a case of predicate ellipsis.

The sentence is an objective subordinate clause lacking the predicate of its governing clause ("[Catiline promised that]"). In ATSs, this is represented by assigning the analytical function ExD (External Dependency) to the main predicate of the sentence. In the ATS of Figure 5, the node for *facturum* is assigned ExD, because here *facturum* depends on a node that is missing and, thus, it is "external" to the current tree.[10]

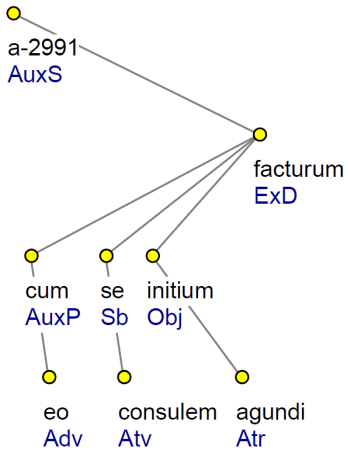


Figure 5. ATS of the sentence "cum eo se consulem initium agundi facturum" (BC 21.4).

Figure 6 shows the TGTS for this sentence. The TGTS resolves the ellipsis of the main clause. Three sentences before this one in the text, Sallust writes "Catiline polliceri" ("Catiline promised [to men]"). The sentence in BC 21.4 still depends on this clause. Once resolved the ellipsis of *polliceor*, the TGTS must represent its arguments. Among these, both the Actor and the Addresse result from ellipsis resolution: Catiline is the Actor and the men (*homo*) are the Addresse. The Patient of *polliceor*, instead, is represented by the entire objective subordinate clause of BC 21.4. In this clause, the Actor is again Catiline, as it is represented by the textual coreference of the node depending on *facio* which is assigned t-lemma #PersPron: this node is not newly added because it is textually represented by the reflexive pronoun *se*. The Patient of *facio* is *initium*, which is specified by a restrictor (RSTR; the verb *ago*) governing a newly added node for a General Actor (#Gen). Such Actor is assigned when its denotation cannot be retrieved contextually, which mostly happens when impersonal clauses are concerned, like in this case (literaly: "the beginning of acting").

The prepositional phrase "cum eo" ("with him") is represented in the TGTS of Figure 6 by the node for *is* (form *eo*), while that for the preposition *cum* collapses. The personal pronoun *is* is linked with a previous occurrence of the proper name *Antonius* via a textual coreference and it is assigned functor ACMP, which is used for the adjuncts that express manner by specifying a circumstance (an object, person,

[10]The analytical function Atv is assigned to verbal attributes, i.e. (predicative) complements not participating in government (*consulem*). AuxP is used for prepositions (*cum*).

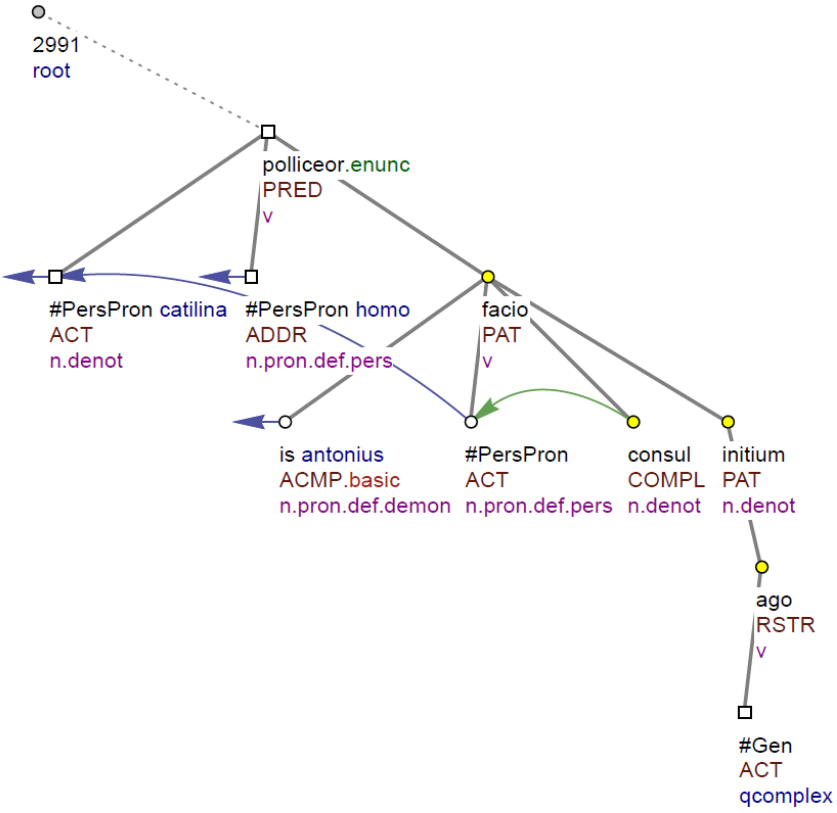


Figure 6. TGTS of the sentence "cum eo se consulem initium agundi facturum" (BC 21.4).

event) that accompanies (or fails to accompany) the event or entity modified by the adjunct.

In TGTSs, predicative complements (functor: COMPL) are adjuncts with a dual semantic dependency relation. They simultaneously modify a noun and a verb. The dependency on the verb is represented by means of an edge. In Figure 6, this is the edge that connects *facio* with *consul*. The dependency on the noun is represented by means of a specific complement reference, which is graphically represented by an arrow (going from *consul* to #PersPron in Figure 6).

3. Methodology

One of the added values of tectogrammatical annotation is that it provides information that, although it is accessible to readers, is explicitly missing in the text. For instance, looking at the example sentence whose ATS and TGTS are shown in Figures 5 and 6 respectively, we see that there is no explicit occurrence of Catiline playing the role of Actor of a verb. Instead, if we exploit tectogrammatical annotation, we can retrieve that actually that sentence carries (implicit) information about the fact that Catiline performs two different Actions (namely, *polliceor* and *facio*).

Tectogrammatical annotation puts us in the condition to answer the basic research question of the work described in this paper: "who does what in *Bellum Catilinae*?". In other words, what we look for are all the couples Actor-Action in BC regardless of the fact that they do explicitly occur in the text.[11]

3.1. Querying the Data

All data can be freely downloaded from the website of the IT-TB project. The treebanks can be queried through an implementation of the PML-TQ search engine (Prague Markup Language – Tree Query) (Štěpánek and Pajas, 2010). We ran a bunch of queries in order to retrieve all the couples Actor-Action in BC. The basic query just searches for all the Actors of a verb:

```
t-node $n0 := [ gram/sempos = 'v',
echild t-node $n1 := [ functor = 'ACT' ] ];
```

This query searches for all the nodes of a TGTS (`t-node`, named `$n0`) that are assigned PoS verb (`gram/sempos = v'`) and govern either directly or indirectly (`echild`) a t-node (`$n1`) with functor ACT (`functor = 'ACT'`).[12] The query does not limit the output to nodes with an explicit textual correspondence, but includes also those newly added in TGTSs, as result of ellipsis resolution.

The output resulting from the query above needs further refinement, as it features several cases of both relative and personal pronouns whose denotation is resolved in TGTSs by coreference analysis. For instance, three Actor-Action couples result from the TGTS of Figure 6: #PersPron-*polliceor*, #PersPron-*facio* and #Gen-*ago*. While #Gen is a General argument whose denotation cannot be retrieved contextually, both the #PersPron nodes are assigned a textual coreference in the TGTS, thus enabling to replace them with the t-lemma they are coreferent with. In particular, the newly added

[11] In this work, we consider Actions as represented by verbs only. Deverbal nominalizations are thus excluded.

[12] Direct or indirect government is set in order to retrieve Actors occurring in coordinated constructions (headed by the coordinating element).

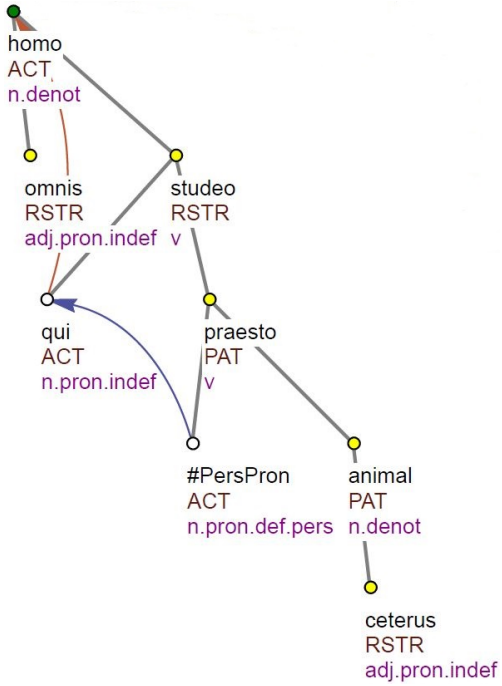


Figure 7. TGTS of the phrase "Omnis homines, qui sese student praestare ceteris animalibus [...]" (BC 1.1).

#PersPron node depending on *polliceor* is directly linked via textual coreference with its antecedent (*catilina*), while the #PersPron node depending on *facio* shows an indirect linking with its antecedent by passing through the other #PersPron node.

We ran a number of queries to replace in the output of the basic query all coreferred #PersPron t-lemmas with those of the nodes they are linked with via textual coreference. Then we did the same for all coreferred t-lemmas of relative pronouns, which are linked with their antecedent via grammatical coreference.

Not only such queries must consider both direct and indirect linking, as well as textual and grammatical coreference, but they also have to address mixed indirect coreferences. For instance, this is the case of the first noun phrase in the first sentence of BC: "Omnis [all] homines [men], qui [who] sese [themselves] student [be eager] praestare [to stand out] ceteris [others] animalibus [animals] [...]" (BC 1.1) ("All humans who are keen to surpass other animals [...]"). Figure 7 shows the portion of the TGTS for the first sentence of BC concerning this phrase.

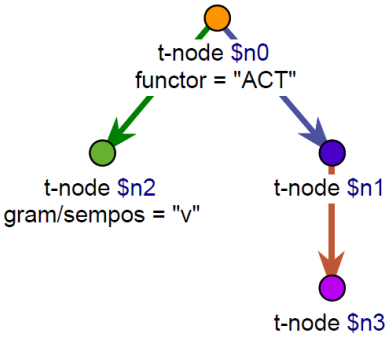


Figure 8. A graphical query in PML-TQ.

From Figure 7, one can see that the denotation (*homo*) of the #PersPron node playing the role of Actor of *praesto* is retrieved (a) indirectly, by passing through the node for *qui*, and (b) in mixed fashion, i.e. via a textual coreference (from #PersPron to *qui*) plus a grammatical coreference (from *qui* to *homo*).

A model of such kind of complex queries is the following (graphically represented in Figure 8):

```
                t-node $n0 := [ functor = 'ACT',
          eparent t-node $n2 := [ gram/sempos = 'v' ],
coref_text.rf t-node $n1 := [ coref_gram.rf t-node $n3 := [ ] ] ];
```

The t-node named `$n0` is an Actor that depends either directly or indirectly (`eparent`) on t-node `$n2`, which is a verb. `$n0` has a textual coreference with `$n1`, which in turn has a grammatical coreference with `$n3`. In the TGTS of Figure 7, `$n2` is the node for *praesto*, `$n0` is the #PersPron node depending on *praesto*, `$n1` is *qui* and `$n3` is *homo*. By just printing in the output of the query the t-lemma for node `$n3`, it is possible to replace #PersPron with *homo* in the list of the Actor-Action couples.[13]

3.2. Networking the Data

Once built the list of all the Actor-Action couples and having enhanced each couple with its frequency of occurrence in the TGTSs of BC, we induced automatically a network from the list.

In order to build the network out of the tectogrammatical annotation of BC, we applied the method developed by Ferrer i Cancho et al. (2004). According to this

[13]The longest coreference chain we found in BC includes 5 textual coreferences.

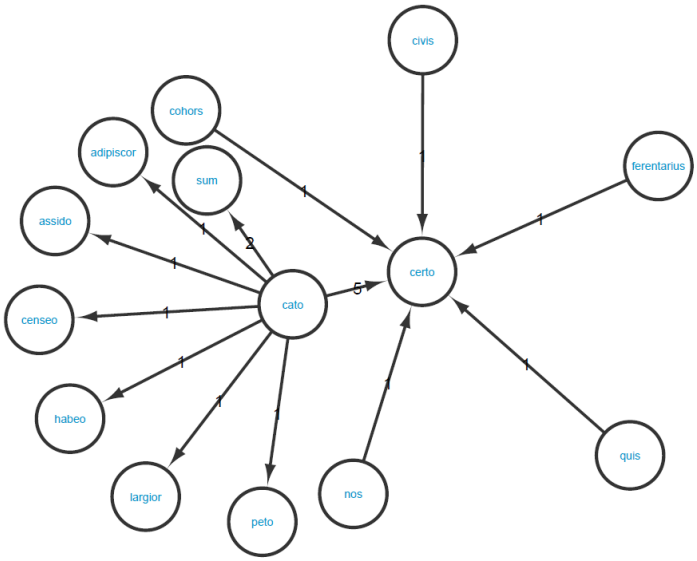


Figure 9. The tecto-based subnetwork for cato and certo.

method, a dependency relation appearing in the source treebank is converted into an edge in the network and two vertices are linked in the network if they appear at least once in a dependency relation in the treebank. The edges are directed according to the direction of the dependency relation in the treebank. In the case of our network, the vertices are Actors and Actions, and the edges are the dependency relations holding between Actors and Actions. The edges are directed from the Actors to the Actions.

The network is built by accumulating sentence structures from the treebank. The treebank is parsed sentence by sentence and new vertices are added to the network. When a vertex is already present in the network, more links are added to it.

The result is a "tecto-based" network containing all the dependencies between Actors and Actions in the input treebank. Edges are weighted by frequency, i.e. each connection between two vertices is enhanced with the number of its occurrences in the source TGTSs.

Figure 9 shows the portion of the tecto-based network concerning the Actor *cato* and the Action *certo* 'to fight'. The node for *cato* is connected to all the Actions that *cato* performs in BC via outgoing edges enhanced with the frequency of the connection they represent; for instance, from Figure 9 one can understand that *cato* performs the Action represented by *certo* five times. Conversely, the node for *certo* is connected to all the Actors that perform such Action via ingoing edges.

The full tecto-based network of BC is shown in Figure 10. The nodes of this network represent all the Actors and the Actions of BC, while its edges are all the dependency relations holding between them in the source TGTSs.

In the following, we first use some topological properties of the tecto-based network of BC to study Actors and Actions in BC. Then, we run a clustering analysis of its vertices with the highest out-degree (i.e. the Actors reported in Table 1) to understand if they can be properly organized into homogeneous groups defined by the set of the vertices they are connected to via outgoing edges (i.e. the Actions they perform).

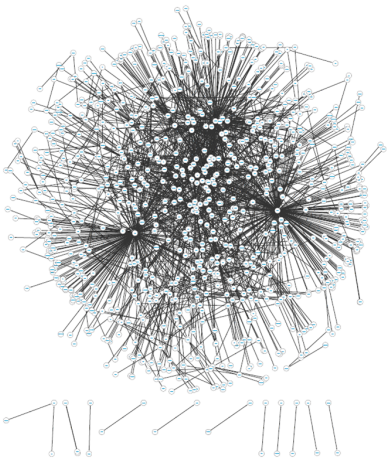


Figure 10. The tecto-based network of Bellum Catilinae.

4. Results and Discussion

4.1. Actors and Actions

Table 1 reports the main Actions and the main Actors in BC. These are defined as the vertices in the tecto-based network with the highest out-degree (Actors) and in-degree (Actions) respectively.[14] In other words, this means that the main Actors are those that perform the highest number of different Actions and, conversely, the main Actions are those performed by the highest number of different Actors.[15]

Beside Actions and the number of their different Actors, Table 1 reports also the total number of occurrences of each Action and, among these, the number of generated occurrences (resulting from ellipsis resolution). The case of *convenio* 'to come together' is worth noting, as it turns out that it has 20 different Actors for just 8 occurrences (2 of which are generated). This happens because in some of its occurrences *convenio* has more than one Actor, like for instance in the sentence "eo [there] convenere [to come together] senatorii [senatorial] ordinis [order] P. Lentulus Sura , P.

[14]In a network, the degree of a vertex *s* is the number of its edges, i.e. different relations holding between *s* and other vertices in the network. In a directed network (like the tecto-based network here concerned), the degree results from the sum of the out-degree, which labels the number of edges that are directed from the vertex, and of the in-degree, which labels the number of edges that are directed to the vertex.

[15]The absence of verbs like *possum* 'can' and *volo*, *velle* 'to want' in Table 1 is due to the treatment of modal predicates in TGTS (see Mikulová, 2006, pp. 318–320). Not coreferred Actors are excluded from Table 1. These are the General Actor (#Gen) and those pronouns that do not undergo coreference analysis in TGTSs, i.e. indefinite and interrogative pronouns (like *alius* and *quis*), as well as both explicit and generated personal pronouns of first and second person.

Action	Actors	Occ.	Generated
sum	179	268	38
habeo	43	84	10
facio	39	87	4
convenio	20	8	2
dico	18	41	9
do	18	22	3
hortor	16	11	2
venio	14	11	0
coepio	13	18	7
puto	13	10	0
peto	13	12	0
cognosco	13	20	0

Actor	Actions	Occ.	Generated
catilina	133	61	6
cicero	33	18	0
homo	32	40	3
res	24	147	4
petreius	20	3	0
lentulus	20	27	6
consul	20	32	0
caesar	20	13	0
populus	19	18	0
curius	19	5	0
vulturcius	18	10	0
vir	18	16	0
animus	18	59	2

Table 1. Main Actions (left) and Actors (right).

Autronius , L. Cassius Longinus , C. Cethegus , P. et Ser . Sullae Ser. filii , L. Vargunteius , Q. Annius , M. Porcius Laeca , L. Bestia , Q. Curius" (BC 17.3) ("There were present from the senatorial order [...]").

Not surprisingly, Catiline is the star of BC, being the Actor of 133 different Actions (i.e. verbs) in 61 occurrences (6 out of which are generated). Traditionally, together with Catiline, the three other main characters of BC are considered to be Caesar, Cato and Cicero, who give the main speeches reported in the text. If we look at the Actions each of them performs and focus on those that Catiline only performs (i.e. those not shared with the others), we can see which Actions are peculiar of Catiline. These are represented by the verbs *dimitto* 'to send out' and *paro* 'to prepare'.

Interestingly enough, *dimitto* and *paro* not only correspond to the Actions performed by Catiline only (and not also by Caesar, Cato or Cicero), but they are also those Actions that Catiline most frequently performs (6 times), just after *facio* 'to make' (10) and *habeo* 'to have' (7), and more than *sum* 'to be' (5) and *video* 'to see' (5). If for *dimitto* this result is biased by a case of ellipsis resolution applied on a multiple coordination in one sentence,[16] *paro* offers a wider range of occurrences. By exploiting

[16] "Igitur [Catilina] C. Manlium Faesulas atque in eam partem Etruriae [dimisit], Septimium quendam Camertem in agrum Picenum [dimisit], C. Iulium in Apuliam dimisit, praeterea alium alio [dimisit], quem ubique opportunum sibi fore credebat" (BC 27.1) ("He, therefore, dispatched Gaius Manlius to Faesulae and that region of Etruria, a certain Septimius of Camerinum to the Picene district, and Gaius Julius to Apulia; others too to other places, wherever he believed that each would be serviceable to him"). The three occurrences of *dimisit* put in square brackets are generated in the TGTS of this sentence via ellipsis resolution. *Catilina* is generated as well, playing the role of Actor of all the generated occurrences of *dimisit*.

semantic role labeling, we can know what Catiline prepares in BC. The most frequent Patients of the occurrences of *paro* in BC with Catiline as Actor are the following: *arma* 'implements of war' 'weapons', *incendium* 'burning', *insidiae* 'trap' and *interficio* 'to destroy'. Such Patients of *paro* show the complexity of the character of Catiline, who is depicted somewhere negatively (mostly in the first half of BC) and somewhere else positively. In fact, while looking at the Patients of *paro*, we see that Catiline is not only someone who prepares malitious acts (*insidias parare*), but he also encourages the revolutionaries to the arms (*arma parare*), which is presented by Sallust under a positive light, as Wilkins (1994) points out (page 51).

Given that Catiline plays the role of Actor in BC more than three times more than Cicero, one can expect that most of the Actions performed by Cicero are common with Catiline and that these Actions are more frequently performed by Catiline than by Cicero. Actually, there are some deviations from such trend. The most clear example is the verb *refero* 'to bear back' 'to report', whose Actor is Cicero in two occurrences while Catiline does never perform it. Moreover, there are three verbs that feature Cicero as Actor more than once and more than Catiline. These are *cognosco* 'to know' and *praecipio* 'to take in advance' 'to warn'. Both these verbs have Cicero as Actor twice and Catiline once. Finally, the Action most frequently performed by Cicero (3) is represented by the verb *iubeo* 'to give an order' 'to command'. Also Catiline is Actor of *iubeo*, but only in two occurrences.

4.2. Clustering the Actors

Clustering is a technique that deals with finding a structure in a collection of data. In particular, clustering is the process of organizing objects (called "observations") into groups ("clusters") whose members are similar in some way. One of trickiest issues in clustering is to define what 'similarity' means and to find a clustering algorithm that computes efficiently the degree of similarity between two objects that are being compared.

Hierarchical clustering is a specific method of cluster analysis that seeks to build a hierarchy of clusters. Hierarchical clustering can be performed by following two main strategies: (a) agglomerative (bottom-up): each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy; (b) divisive (top-down): all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In this work, we apply hierarchical agglomerative clustering to compute the degree of similarity/dissimilarity between the Actors reported in Table 1. Such degree is obtained by comparing Actors by the Actions they perform. First, we compute the amount of shared and non-shared Actions between the members of all the possible couples of Actors. Then, we compare the distribution of shared and not shared Ac-

tions by their relative frequency.[17] As for the distance measure, the analysis is run on document-term matrices by using the cosine distance[18]

$$d(i; i') = 1 - \cos\{(x_{i1}, x_{i2}, ..., x_{ik}), (x_{i'1}, x_{i'2}, ..., x_{i'k})\} .$$

The arguments of the *cosine* function in the preceding relationship are two rows, i and i', in a document-term matrix; x_{ij} and $x_{i'j}$ provide the number of occurrences of verb j (j =1, ..., k) in the two sets of Actions corresponding to rows i and i' ("profiles"). Zero distance between two sets (cosine = 1) holds when two sets with the same profile are concerned (i.e. they have the same relative conditional distributions of terms). In the opposite case, if two sets do not share any word, the corresponding profiles have maximum distance (cosine = 0).

As for clustering, we run a "complete" linkage agglomeration method. While building clusters by agglomeration, at each stage the distance (similarity) between clusters is determined by the distance (similarity) between the two elements, one from each cluster, that are most distant. Thus, complete linkage ensures that all items in a cluster are within some maximum distance (or minimum similarity) to each other.

Roughly speaking, according to our clustering method, Actors that share a high number of Actions with similar distribution are considered to have a high degree of similarity and, thus, fall into the same or related clusters. Figure 11 plots the results.

Looking at Figure 11, it turns out that there are three main clusters.

Moving from top to bottom, the first cluster includes the two most similar Actors according to the Actions they perform. These are *cicero* and *consul* 'consul'. This happens although BC includes several occurrences of *consul* that are not referred to Cicero. Actually, Marcus Tullius Cicero is the consul par excellence in Roman political history and he was the only consul among the Actors considered here, as Caesar would become consul for the first time in 59 BCE, four years after the facts told in BC. The second most similar couple of Actors is the one including *catilina* and *lentulus*. Catiline was the one who devised the conspiracy narrated in BC. Publius Cornelius Lentulus was one of the main conspirators. In particular, he took the place of Catiline as chief of the conspirators in Rome, when Catiline had to leave the city after the famous second speech of Cicero *In Catilinam*. The two characters are, thus, strictly related. This is further confirmed by the following words of Cato's speech, which closely connect the decision to be taken by the Senate about Lentulus with that about the army of Catiline: "Qua re quom de P. Lentulo ceterisque statuetis, pro certo habetote vos simul de exercitu Catilinae et de omnibus coniuratis decernere" (BC 52.17)

[17] All the experiments were performed with the R statistical software (R Development Core Team, 2012). More details about the clustering method used here can be found in Passarotti and Cantaluppi (2016).

[18] A document-term matrix is a mathematical matrix that holds frequencies of distinct terms for each document. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

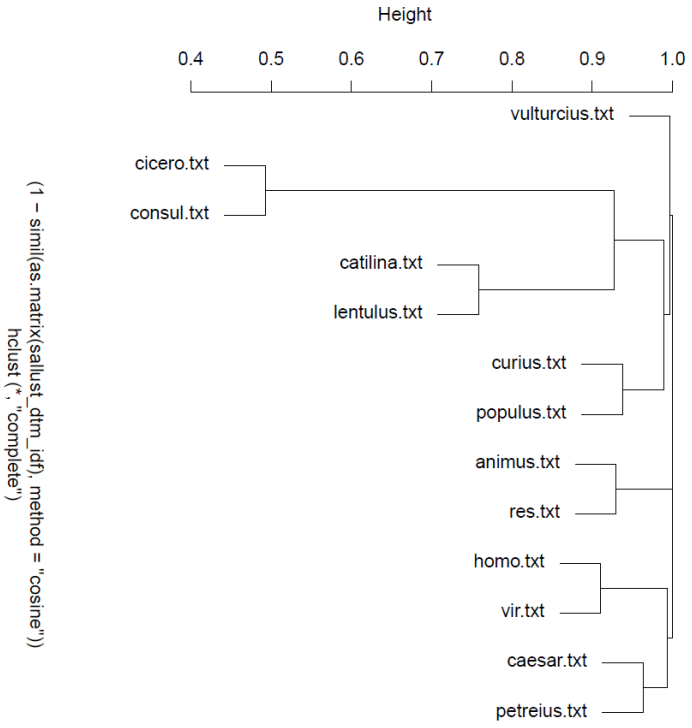


Figure 11. Clustering the Actors.

("Be assured, then, that when you decide the fate of Publius Lentulus and the rest, you will at the same time be passing judgment on Catiline's army and all the conspirators"). In this respect, the destinies of Lentulus and Catiline are not only linked to each other, but they are also strictly bound to the outcome of the conspiracy. Indeed, as Wilkins (1994; page 95) points out, Lentulus's execution on one side and the death of Catiline on the other represent respectively the first and the last step in the failure of the conspiracy.

In the same larger cluster are *curius* and *populus* 'people'. Quintus Curius was another conspirator, although his role was actually ambivalent. Being a friend of Catiline, he took part in the conspiracy, but at the same time it was because of him that it was foiled. According to Sallust, Curius, to boast with his mistress Fulvia, told her the details of the conspiracy, which she informed Cicero about. Moreover, Curius accused Caesar of being a conspirator. Such an undefined role is played also by "the people". In those passages where Sallust talks about "the Roman people" 'populus romanus', these are mostly positively depicted. Conversely, there are also places in

BC where the people act badly. Finally, Titus Vulturcius, a conspirator playing a subordinate role in the plot, falls into the same cluster, standing quite apart from the others.

The second cluster includes just two lemmas: *animus* 'soul' and *res* 'thing'. These are the only not human Actors, among the ones considered here.

The third cluster features two couples of Actors. The first includes lemmas *homo* 'human being' 'man' and *vir* 'adult male' 'man', which are semantically strictly related, standing in hypernym/hyponym relation. The second couple is formed by *petreius* and *caesar*. Marcus Petreius plays a positive role in BC, having led the senatorial forces in the victory over Catiline in Pistoia. It is worth noting that such a positive character in the plot gets clustered together with Caesar. The future dictator Gaius Iulius Caesar hoped for the success of the second conspiracy of Catiline, just like he did for the first. However, Sallust's intent is to lift Caesar of any suspicion of a possible link with Catiline. He emphasizes the Caesar's concern for legality, depicting him (together with Cato) as the faithful guardian of "mos maiorum", the core, unwritten code of Roman traditionalism. Putting Caesar under such a positive light is strictly connected to the fact that, while BC was being written, Caesar was deified by decree of the Roman Senate (on 1st January 42 BCE), after his assassin on the Ides of March 44 BCE.

5. Conclusion and Future Work

In the context of the work presented in this paper, there are two main open issues to address in the near future. First, we must enhance data with coreference analysis of either explicit or generated first and second personal pronouns. This is needed because BC features speeches given by different characters, which makes the reference of such pronouns change across the text. Second, our work must consider also Actions represented by (deverbal) nouns together with their either explicit or generated Actors, which tend to occur as subjective genitives.

As far as the interpretation of some specific aspects of BC is concerned, we have shown that using tectogrammatical annotation for studying the Actors and the Actions of the plot can clarify to some extent how Sallust conceived Catiline's portrayal. In this respect, future work must focus specifically on the Actions performed by Catiline, by taking into consideration the structure of the book in order to determine the evolution of the character throughout the chapters. Also, performing the tectogrammatical annotation of Cicero's *In Catilinam* would help to compare the two portrayals of Catiline.

More generally speaking, the work described here represents a case study showing how much useful a treebank enhanced with semantic annotation can be for literary studies. In this respect, there is still much to do. On one side, still too few literary texts provided with such annotation layer are currently available. On the other, the use of linguistic resources like treebanks remains dramatically confined in the area

of computational and theoretical linguistics, not impacting other communities which might largely benefit from such resources.

To overcome the former, one desideratum is building NLP tools able to provide good accuracy rates of semantic annotation across different domains. As for the latter, developers of treebanks based on literary data and/or texts written in ancient languages must more and more get in touch with different kinds of domain experts from the Humanities, like philologists, historical linguists, philosophers, historians and scholars in literature. Indeed, across the last few years, this looks like a growing trend, with several events and special issues of scientific journals dedicated to different topics in computational linguistics and the Humanities. We hope that this is just the beginning of a fruitful joint work.

Acknowledgements

This research is supported by the Italian Ministry of Education, University and Research (MIUR), FIR-2013 project "Developing and Integrating Advanced Language Resources for Latin" (ID: RBFR13EWQN).

Bibliography

Ahlberg, Axel W. *C. Sallusti Crispi. Catiline, Iugurtha, Orationes Et Epistulae Excerptae De Historiis*. Teubner, Leipzig, 1919.

Bamman, David and Gregory Crane. The Ancient Greek and Latin Dependency Treebanks. In *Language Technology for Cultural Heritage*, pages 79–98. Springer, 2011. URL https://doi.org/10.1007/978-3-642-20227-8_5.

Bamman, David, Marco Passarotti, Gregory Crane, and Savina Raynaud. *Guidelines for the Syntactic Annotation of Latin Treebanks*. Tufts University Digital Library, Boston, MA, 2007. URL https://itreebank.marginalia.it/doc/2007_Passa+Bamman+Crane+Raynaud_Guidelines%20Tb.pdf.

Batstone, William. Word at War: The Prequel. In *Citizens of Discord: Rome and Its Civil Wars*, pages 45–72. OUP USA, 2010.

Broughton, Thomas RS. Was Sallust Fair to Cicero? In *Transactions and Proceedings of the American Philological Association*, pages 34–46. JSTOR, 1936.

Conley, Duane F. The Interpretation of Sallust Catiline 10. 1-11. 3. *Classical Philology*, 76(2): 121–125, 1981.

Demske, Ulrike. Das MERCURIUS-Projekt: Eine Baumbank für das Frühneuhochdeutsche. *Sprachkorpora: Datenmengen und Erkenntnisfortschritt*, pages 91–104, 2007. URL https://doi.org/10.1515/9783110439083-007.

Earl, Donald C. *The political thought of Sallust*. PhD thesis, University of Cambridge, 1958.

Eckhoff, Hanne Martine and Aleksandrs Berdicevskis. Linguistics vs. digital editions: The Tromsø Old Russian and OCS Treebank. *Scripta & e-Scripta*, 14:15, 2015.

Ferrer i Cancho, Ramon, Ricard V Solé, and Reinhard Köhler. Patterns in syntactic dependency networks. *Physical Review E*, 69(5):051915, 2004. URL https://doi.org/10.1103/physreve.69.051915.

González Saavedra, Berta and Marco Passarotti. Challenges in enhancing the Index Thomisticus treebank with semantic and pragmatic annotation. In *Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT-13). Department of Linguistics, University of Tübingen*, pages 265–270, 2014. URL http://tlt13.sfs.uni-tuebingen.de/tlt13-proceedings.pdf#page=273.

Hajič, Jan, Alena Böhmová, Eva Hajičová, and Barbora Vidová Hladká. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Kluwer, 2000.

Haug, Dag and Marius Jøhndal. Creating a parallel treebank of the old Indo-European Bible translations. In *Proceedings of the Language Technology for Cultural Heritage Data Workshop (LaTeCH 2008)*, pages 27–34. ELRA, 2008. URL http://www.lrec-conf.org/proceedings/lrec2008/workshops/W22_Proceedings.pdf#page=31.

Inglese, Guglielmo. Towards a Hittite Treebank. Basic Challenges and Methodological Remarks. In *Proceedings of the Workshop on Corpus-Based Research in the Humanities (CRH)*, pages 59–68. Institute of Computer Science of the Polish Academy of Sciences, 2015.

Kroch, Anthony, Beatrice Santorini, and Lauren Delfs. The Penn-Helsinki Parsed Corpus of Early Modern English (PPCEME). Department of Linguistics, University of Pennsylvania. CD-ROM. *Department of Linguistics, University of Pennsylvania, CD-ROM*, 2004.

Martineau, France. Un corpus pour l'analyse de la variation et du changement linguistique. *Corpus*, 7, 2008.

McConaghy, Mary Lee Sivess. *Sallust and the Literary Portrayal of Character*. UMI, Washington, 1974.

Meyer, Roland. New wine in old wineskins?—Tagging Old Russian via annotation projection from modern translations. *Russian linguistics*, 35(2):267–281, 2011. URL https://doi.org/10.1007/s11185-011-9075-x.

Mikulová, Marie et al. *Annotation on the Tectogrammatical Layer in the Prague Dependency Treebank*. Institute of Formal and Applied Linguistics, Prague, Czech Republic, 2006. URL https://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/t-layer/pdf/t-man-en.pdf.

Passarotti, Marco. Language Resources. The State of the Art of Latin and the Index Thomisticus Treebank Project. In *Corpus anciens et Bases de données*, pages 301–320. Presses universitaires de Nancy, 2011.

Passarotti, Marco and Gabriele Cantaluppi. A Statistical Investigation into the Corpus of Seneca. In *Latinitatis Rationes. Descriptive and Historical Accounts for the Latin Language*, pages 684–706. De Gruyter, 2016.

Passarotti, Marco and Berta González Saavedra. The Treebanked Conspiracy. Actors and Actions in Bellum Catilinae. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 18–26, 2017. URL http://www.aclweb.org/anthology/W17-7605.

Passarotti, Marco, Berta González Saavedra, and Christophe Onambele. Latin Vallex. A Treebank-based Semantic Valency Lexicon for Latin. In *LREC*, 2016. URL http://www.lrec-conf.org/proceedings/lrec2016/pdf/96_Paper.pdf.

R Development Core Team. *R: A language and environment for statistical computing*. Foundation for Statistical Computing, Vienna, Austria, 2012.

Ramsey, John T. *Sallust. The war with Catiline. The war with Jugurtha*. Harvard University Press, The Loeb Classical Library 116, Cambridge, MA, 2014.

Schröder, Wilt Aden. Zu Sallust, Catilina 3, 3 (und zum Gedankengang des Proömiums). In *Lemmata: Beiträge zum Gedenken an Christos Theodoridis*, pages 203–219. Walter de Gruyter GmbH & Co KG, 2015.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. D. Reidel, Dordrecht, NL, 1986.

Štěpánek, Jan and Petr Pajas. Querying Diverse Treebanks in a Uniform Way. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC 2010)*, pages 1828–1835. ELRA, 2010. URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/381_Paper.pdf.

Syme, Ronald. *Sallust*. University of California Press, Berkeley, 1964.

Tannenbaum, RF. What Caesar Said: Rhetoric and History in Sallust's Coniuratio Catilinae 51. In *Roman Crossings: Theory and Practice in the Roman Republic*, pages 209–223. Classical Press of Wales, 2005.

Taylor, Ann. The York—Toronto—Helsinki parsed corpus of old english prose. In *Creating and digitizing language corpora*, pages 196–227. Springer, 2007. URL https://doi.org/10.1057/9780230223202_9.

Taylor, Ann and Anthony S Kroch. The Penn-Helsinki Parsed Corpus of Middle English. *MS. University of Pennsylvania*, 1994. URL http://www.ling.upenn.edu/mideng/documentation/manual.ps.

Urešová, Zdeňka. Building the PDT-VALLEX valency lexicon. In *On-line proceedings of the fifth Corpus Linguistics Conference. University of Liverpool*, 2009. URL http://ufal.ms.mff.cuni.cz/pcedt2.0/publications/Uresova2011.pdf.

Waters, Kenneth H. Cicero, Sallust and Catiline. *Historia: Zeitschrift für Alte Geschichte*, H. 2: 195–215, 1970.

Wilkins, Ann Thomas. *Villain or hero: Sallust's portrayal of Catiline*, volume 15. Peter Lang Pub Inc, 1994.

Žabokrtský, Zdeněk. Treex – an open-source framework for natural language processing. In *Information Technologies – Applications and Theory*, pages 7–14. Univerzita Pavla Jozefa Šafárika v Košiciach, 2011. URL http://ceur-ws.org/Vol-788/paper2.pdf.

Address for correspondence:
Marco Passarotti
marco.passarotti@unicatt.it
Università Cattolica del Sacro Cuore. Largo Gemelli, 1 - 20123 Milan, Italy

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 29–56

Enriching VALLEX with Light Verbs: From Theory to Data and Back Again

Václava Kettnerová, Markéta Lopatková, Eduard Bejček, Petra Barančíková

Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, Prague, Czechia

Abstract

This paper summarizes results of a theoretical analysis of syntactic behavior of Czech light verb constructions and their verification in the linguistic annotation of a large amount of these constructions. The concept of LVCs is based on the observation that nouns denoting actions, states, or properties have a strong tendency to select semantically underspecified verbs, which leads to a specific rearrangement of valency complementations of both nouns and verbs in the syntactic structure. On the basis of the description of deep and surface syntactic properties of LVCs, a formal model of their lexicographic representation is proposed here. In addition, the resulting data annotation, capturing almost 1,500 LVCs, is described in detail. This annotation has been integrated in a new version of the VALLEX lexicon, release 3.5.

1. Introduction

Light verb constructions (LVCs) pose a serious challenge for both theoretical linguistics and NLP tasks due to their syntactic complexity. The major challenges raised by LVCs can be overcome by a lexicographic representation allowing for their efficient handling in both theoretical and computational linguistics. Developing a formal model of such representation thus represents a crucial task of the current lexicography.

In this paper, we present a formal model of the lexicographic description of LVCs designed for the valency lexicon of Czech verbs VALLEX, summarizing findings partially presented esp. in Kettnerová and Lopatková (2015); Kettnerová et al. (2016); Kettnerová (2017); Kettnerová and Lopatková (2017b), and in Kettnerová and Lopatková

 Corresponding author: kettnerova@ufal.mff.cuni.cz
Cite as: Václava Kettnerová, Markéta Lopatková, Eduard Bejček, Petra Barančíková. Enriching VALLEX with Light Verbs: From Theory to Data and Back Again. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 29–56. doi: 10.2478/pralin-2018-0006.

(2017a). This model, based on a thorough theoretical research into Czech LVCs and grounded in an in-depth analysis of corpus data using the Functional Generative Description (FGD, see esp. Sgall et al., 1986), has been applied in an extensive annotation of Czech LVCs allowing us to verify theoretical adequacy of the adopted postulates and their further modification. This annotation has been integrated in the VALLEX lexicon, release 3.5.

LVCs represent a type of complex predicates where two syntactic elements – a light verb and a predicative noun, adjective, adverb (or verb esp. in Asian languages) – function together as a single predicative unit. The syntactic structure of an LVC is then determined by both the light verb and the other predicative element. For example, the structure with the light verb *dát* 'to give' is determined not solely by the verb, which provides three valency complementations, namely ACT (*matka* 'mother'), ADDR (*děti* 'children'), and CPHR (*příkaz* 'order'), see example (1), but also by the noun *příkaz* 'order', which contributes PAT (*uklidit* 'to clean up') to the structure.[1]

(1) *Matka*$_{\text{ACT}}$ *dala dětem*$_{\text{ADDR}}$ *příkaz*$_{\text{CPHR}}$ *uklidit*$_{\text{PAT}}$ *pokoj.*
'Mother gave children an order to clean their room up.'

The question how valency complementations of light verbs and predicative elements participate in the syntactic structure formation of LVCs is a central issue of any syntactic theory attempting to provide their comprehensive analysis. Despite being addressed in many theoretical frameworks, see e.g., argument merger formulated within the Government Binding theory (Grimshaw and Mester, 1988), argument fusion (Butt, 2010) and argument composition within the Lexical-Functional Grammar (Hinrichs et al., 1998), and the study by Alonso Ramos carried out within the Meaning $\leftrightarrow$ Text Theory (Alonso Ramos, 2007), this issue is still far from being clear. The main difficulty in arriving at a more uniform analysis of LVCs lies in the fact that a definitional characterization allowing for their cross-linguistic identification is still missing. In some analyses, LVCs are then mixed up with control constructions, or even with auxiliary verbs – such misinterpretations clearly lead to inconsistent conclusions (see esp. Butt and Geuder, 2001).

In this paper, we aim at contributing to better understanding the syntactic structure formation of Czech LVCs, making use of the theoretical framework of FGD. We limit our focus to LVCs composed of light verbs and predicative nouns expressed as their direct object, as these LVCs are the most central and frequent ones in Czech. We demonstrate that the syntactic formation of these LVCs is compositional, namely that the syntactic structure of an LVC can be inferred from syntactic properties of the predicative noun and the light verb forming the given LVC on a rule basis. Further, we

[1] Compare with the syntactic structure with the full verb *dát* consisting of ACT (*matka* 'mother'), ADDR (*děti* 'children'), and PAT (*čokoláda* 'chocolate'), see the following example:
Matka$_{\text{ACT}}$ *dala dětem*$_{\text{ADDR}}$ *čokoládu*$_{\text{PAT}}$.
'Mother gave children chocolate.'

show that the coreference relation between valency complementations of predicative nouns and light verbs represents a key characteristic of LVCs.

As has been already reflected in the literature (see e.g., Radimský, 2010), an intuitive clue lying in paraphrasability of LVCs by single verbs does not represent a reliable criterion for their identification as there is a number of broadly accepted LVCs without a suitable paraphrase (e.g., *udělat dojem* 'to make an impression', *nést vinu* 'to be at fault', *mít názor* 'to have an opinion'); on the contrary, some full verb collocations can be paraphrased by single verbs (e.g., *uložit do hrobu* 'to lay to rest; lit. to lay into the grave' – *pohřbít* 'to tomb'). Therefore we consider the syntactic compositionality together with the coreference relation between verbal and nominal valency complementations as a definitional criterion for delimiting LVCs in our research.

The paper is structured as follows. First, the valency lexicon of Czech verbs, VALLEX, and its theoretical background – the valency theory of FGD – are introduced (Section 2). Second, the results of the theoretical analysis of Czech LVCs are summarized (Section 3). On the basis of these results, a formal model of the lexicographic representation of LVCs is proposed (Section 4). Third, the application of this model to the annotation of a large amount of linguistic data is described and the resulting annotated data are characterized (Section 5). The last section provides a short summary.

2. VALLEX and the Valency Theory of FGD

VALLEX, a valency lexicon of Czech verbs, takes the Functional Generative Description (FGD) as its theoretical background. FGD represents a dependency oriented framework which adopts a stratificational approach to the language description (Sgall et al., 1986). One of the main concept of FGD is represented by the tectogrammatical layer – the deep syntactic layer. The core of this layer is represented by valency, an ability of a word to open a certain number of valency positions for other dependent units. The valency theory of FGD has been elaborated from the 70th esp. by Jarmila Panevová (see esp. Panevová, 1974–75, 1980, 1994) and applied in several valency lexicons: *VALLEX*, a valency lexicon of Czech verbs (Lopatková et al., 2016),[2] *PDT-Vallex*, a valency lexicon linked to the family of Prague Dependency Treebanks (Urešová, 2011),[3] and *EngVallex*, a valency lexicon representing the part of the annotation of the Prague Czech-English Dependency Treebank.[4]

In the valency theory of FGD, actants and free modifications are distinguished. Actants (be they obligatory, or optional) represent valency complementations char-

[2] http://ufal.mff.cuni.cz/vallex (http://hdl.handle.net/11234/1-2307)

[3] http://ufal.mff.cuni.cz/pdt-vallex-valency-lexicon-linked-czech-corpora (http://hdl.handle.net/11858/00-097C-0000-0023-4338-F)

[4] http://ufal.mff.cuni.cz/engvallex-english-valency-lexicon-linked-corpora (http://hdl.handle.net/11858/00-097C-0000-0023-4337-2)

acterizing the word in a unique way; as such they have to be listed in its valency frame. Five actants are recognized: ACT, PAT, ADDR, ORIG, and EFF. A higher number of free modifications is determined on the basis of their semantic features (e.g., temporal, spatial, causal, etc.); only obligatory ones characterize a word in a unique way and thus they have to be part of its valency frame as well (see esp. Panevová, 1994; Lopatková and Panevová, 2006). Morphemic forms of actants are determined by a governing word (as such they have to be indicated in its valency frame) while forms of free modifications stem from their semantic type (as a result, they do not have to be provided in the valency frame) (Lopatková and Panevová, 2006).

The VALLEX lexicon, attempting to provide a comprehensive description of valency behavior of Czech verbs, represents the most elaborated lexicon developed within FGD. For purposes of the description of language phenomena at the lexicon-grammar interface, this lexicon has been divided into two components: a lexical part (the data component), providing information specific to individual lexical units, and a grammar part (the grammatical component), capturing regular patterns of Czech verbs in the form of formal rules (thus being a part of the overall grammar of Czech).

Data component provides the information on valency structure of Czech verbs in their individual senses. The key organizing concept of this lexicon is represented by the *lexeme*, an abstract twofold unit associating lexical forms of a verb with its lexical units (individual senses). Each *lexical unit* is assigned the syntactic and semantic information. VALLEX stores more than 6,760 lexical units of verbs (counting aspectual counterparts separately 10,900 lexical units) contained in more than 2,730 lexemes. These lexemes are represented by almost 4,600 verb lemmas.

The crucial information on the valency structure of individual lexical units is provided in the form of valency frame. Valency frame is modeled as a sequence of valency slots, each slot standing for one valency complementation. Each slot of a valency complementation is characterized by a functor (a syntactico-semantic label marking the relation of the valency complementation to its governing verb, as e.g., ACT, ADDR, PAT, LOC etc.) and by the information on obligatoriness. In addition, the information on morphemic forms is provided for actants, indicating possible surface syntactic expression of the given complementation in active, unreciprocal and irreflexive constructions.

Each lexical unit can be described by other relevant syntactic and syntactico-semantic information, e.g., on control, reflexivity, reciprocity, diatheses, alternations, syntactico-semantic class membership.

Grammar component stores grammatical rules which instantiated by the information provided by the data component of the lexicon allow for obtaining all possible surface syntactic manifestations of lexical units of verbs, namely their passive, reciprocal and reflexive structures.

As we show below, the division into the lexical and grammar component has appeared to be relevant also for the description of LVCs as they represent a typical language phenomenon bridging these two parts of the language description.

3. Theoretical Analysis of LVCs

In this section, we provide a comprehensive theoretical account of both deep (Section 3.1) and surface structure formation of LVCs (Section 3.2), grounded in FGD.

3.1. Deep Syntactic structure of LVCs

The deep syntactic structure of LVCs consists of valency complementations of both predicative nouns (Section 3.1.1) and light verbs (3.1.2). In addition, it is characterized by coreference between valency complementations of light verbs and predicative nouns (Section 3.1.4).

3.1.1. Valency Frames of Predicative Nouns

As predicative nouns, deverbal nouns (e.g., *plán* 'plan', *hněv* 'rage', *nedůvěra* 'distrust', *nenávist* 'hate', *svědectví* 'evidence'), deadjective nouns (e.g., *vděčnost* 'gratitude', *žárlivost* 'jealousy'), and primary nouns (e.g., *láska* 'love') occur. Predicative nouns denote actions, states (deverbal and primary nouns), and properties (deadjective nouns). Deverbal and deadjective predicative nouns typically inherit the valency structure from their base verbs and adjectives, respectively. The valency structure of primary nouns can be accounted for on the basis of the valency structure of verbs with corresponding meanings (e.g., *láska* 'love' and *milovat* 'to love'). The number and type of valency complementations of predicative nouns (i.e., their functors and obligatoriness) thus remain the same (see esp. Kettnerová et al., 2017). Only morphemic forms of their complementations usually undergo changes, reflecting their usage in nominal structures. Compare, e.g., the valency frame of the predicative noun *obava* 'fear' (5) and its nominal structure (7) with the frame of its base verb *obávat se* 'to fear; to be afraid' (2) and its verbal structure (4).

(2) *obávat se* 'to fear; to be afraid': $\mathsf{ACT}_{\mathsf{nom}}\ \mathsf{PAT}_{\mathsf{gen,inf,dcc}}$

(3)
$$\begin{array}{lcl} \mathsf{ACT}_{\mathsf{V}} & \Leftrightarrow & \text{'Experiencer'} \\ \mathsf{PAT}_{\mathsf{V}} & \Leftrightarrow & \text{'Stimulus'} \end{array}$$

(4) *Věřící lidé*$_{\mathsf{ACT.nom}}$ *se méně obávají smrti*$_{\mathsf{PAT.gen}}$.
'Believers$_{\mathsf{ACT}}$ are less afraid of death$_{\mathsf{PAT}}$.'

(5) *obava* 'fear': $\mathsf{ACT}_{\mathsf{gen,pos}}\ \mathsf{PAT}_{\mathsf{před+instr,z+gen,inf,dcc}}$

(6)
$$\begin{array}{lcl} \mathsf{ACT}_{\mathsf{N}} & \Leftrightarrow & \text{'Experiencer'} \\ \mathsf{PAT}_{\mathsf{N}} & \Leftrightarrow & \text{'Stimulus'} \end{array}$$

(7) *obavy věřících lidí*$_{\text{ACT.gen}}$ *ze smrti*$_{\text{PAT.z+gen}}$
'believers'$_{\text{ACT}}$ fear of death$_{\text{PAT}}$'

Each valency complementation of a predicative noun corresponds to a semantic participant of the situation denoted by the noun. Semantic participants of a noun are typically identical to the participants characterizing its base verb (or a semantically corresponding verb). For example, the valency complementations ACT and PAT of the predicative noun *obava* 'fear' are mapped onto the semantic participant 'Experiencer', a sentient entity experiencing fear, and 'Stimulus', evoking the given emotion, respectively, as complementations of its base verb *obávat se* 'to fear; to be afraid', see (6) and (3), respectively.[5]

3.1.2. Valency Frames of Light Verbs

Light verbs, as semantically impoverished verbs (e.g., Jespersen, 1965; Grimshaw and Mester, 1988; Butt and Geuder, 2001), denote only general semantic scenarios. They typically inherit valency characteristics from respective full verb counterparts – while one of their valency position is reserved for predicative nouns (this position is labeled with the CPHR functor in FGD), other valency complementations acquire semantic specifications in LVCs via coreference with valency complementations of predicative nouns (see esp. Alonso Ramos, 2007); Section 3.1.4 discusses this feature in more detail.[6] The only exception represented by 'Causator' is discussed below in this Section.

Let us compare the valency frame of the full verb *mít* 'to have' (8) and the frame of the light verb *mít* 'to have' (10) and their example sentences (9) and (12), respectively. The full verb refers to a possession of an object by an owner (its valency complementations are mapped onto the semantic participants: ACT onto 'Owner' and PAT onto 'Possession'). In contrast, the light verb, denoting a general scenario, does not have any semantic participants semantically specifying its valency complementations. Their complementations are semantically specified just in LVCs, by entering in coreference with nominal complementations. For example, in the LVC *mít obavu* 'to be afraid; lit. to have a fear', the verbal ACT attains its semantic saturation via coreference with the nominal ACT, namely it refers to 'Experiencer', see the valency frame (5) and the mappings (6) and (11).

(8) *mít* 'to have': ACT$_{\text{nom}}$ PAT$_{\text{acc}}$

(9) *Petr*$_{\text{ACT.nom}}$ *má nový dům*$_{\text{PAT.acc}}$.
'Peter$_{\text{ACT}}$ has a new house$_{\text{PAT}}$.'

[5] In examples below, the $\Leftrightarrow$ arrows indicate the mapping between valency complementations and semantic participants; the $\equiv$ sign indicates predicative nouns filling the CPHR position; the $\leftrightarrow$ arrows are reserved for coreferential relations between verbal and nominal valency complementations.

[6] The possible reduction of valency frames of light verbs is discussed in Kettnerová and Lopatková (2013).

(10) *mít* 'to have': $\mathsf{ACT}_{\mathsf{nom}}$ $\mathsf{CPHR}_{\mathsf{acc}}$

(11) *mít obavu* 'to be afraid':

		$\mathsf{ACT}_{\mathsf{V}}$	$\leftrightarrow$	$\mathsf{ACT}_{\mathsf{N}}$	$\Leftrightarrow$ 'Experiencer'
				$\mathsf{PAT}_{\mathsf{N}}$	$\Leftrightarrow$ 'Stimulus'
obava 'fear'	$\equiv$	$\mathsf{CPHR}_{\mathsf{V}}$			

(12) *Petr*$_{\mathsf{ACT.nom}}$ *má ze zkoušky*$_{\mathsf{PAT.z+gen}}$ *obavy*$_{\mathsf{CPHR.acc}}$.
'Peter$_{\mathsf{ACT}}$ is afraid of the exam$_{\mathsf{PAT}}$.'

The only exception when light verbs contribute their semantic participant to LVCs is represented by light verbs with a *causative function* – these verbs contribute the semantic participant 'Causator' to LVCs. This participant instigates the events expressed by predicative nouns with which the given light verbs combine. For example, the light verb *poskytovat* 'to provide' has the causative function in the LVC *poskytovat příležitost* 'to provide an opportunity'. The valency frame of this verb is provided in (13). This light verb provides the given LVC with the 'Causator' which is mapped onto its ACT. This 'Causator' serves as an instigator of the situation expressed by the noun *příležitost* 'opportunity', see mapping (16) and example (17).

(13) *poskytovat* 'to provide': $\mathsf{ACT}_{\mathsf{nom}}$ $\mathsf{ADDR}_{\mathsf{dat}}$ $\mathsf{CPHR}_{\mathsf{acc}}$

(14) *příležitost* 'opportunity': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{gen,k+dat,na+acc,pro+acc,inf,dcc}}$

(15) *naše*$_{\mathsf{ACT.poss}}$ *příležitost zamyslet se*$_{\mathsf{PAT.inf}}$ *nad tím, co je to vina*

(16) *poskytovat příležitost* 'to provide an opportunity':

'Causator'	$\Leftrightarrow$	$\mathsf{ACT}_{\mathsf{V}}$				
		$\mathsf{ADDR}_{\mathsf{V}}$	$\leftrightarrow$	$\mathsf{ACT}_{\mathsf{N}}$	$\Leftrightarrow$	'Agent'
příležitost 'opportunity'	$\equiv$	$\mathsf{CPHR}_{\mathsf{V}}$				
				$\mathsf{PAT}_{\mathsf{N}}$	$\Leftrightarrow$	'Future_action'

(17) *Jeho skutek*$_{\mathsf{ACT.nom}}$ *nám*$_{\mathsf{ADDR.dat}}$ *poskytl příležitost*$_{\mathsf{CPHR.acc}}$ *zamyslet se*$_{\mathsf{PAT.inf}}$ *nad tím, co je to vina.*
'His act$_{\mathsf{ACT}}$ gave us$_{\mathsf{ADDR}}$ an opportunity$_{\mathsf{CPHR}}$ to think$_{\mathsf{PAT}}$ about what is guilt.'

3.1.3. Semantic vs. Syntactic Center of LVCs

The syntactic center of LVCs is represented by the light verb, which can – in contrast to the predicative noun – create a finite clause. However, the semantic core of LVCs is formed by the predicative noun, which contributes its semantic participants to LVCs.

Selecting a particular light verb, the predicative noun can employ its semantic participants in the syntactic structure of a finite clause. Moreover, the choice of a light verb affects the perspective from which the situation expressed by the predicative noun is viewed (see esp. Kettnerová and Lopatková, 2015). Compare, e.g., the LVC *poskytnout dotaci* 'to give a grant' in (18) and the LVC *získat dotaci* 'to obtain a grant' in

(19) where each time a different semantic participant of the noun *dotace* 'grant' occupies the most prominent subject position: the LVC *poskytnout dotaci* 'to give a grant' is perspectivized from the point of view of 'Agent' (*vláda* 'government') whereas the LVC *získat dotaci* 'to obtain a grant' is presented from the perspective of 'Recipient' (*město* 'town'):

(18) *Vláda*$_{\text{ACT}}$ *poskytla městu*$_{\text{ADDR}}$ *dotaci*$_{\text{CPHR}}$ *7 milionů*$_{\text{PAT}}$ *korun.*
'The government$_{\text{ACT}}$ gave a grant$_{\text{CPHR}}$ of 7 million$_{\text{PAT}}$ crowns to the town$_{\text{ADDR}}$.'

(19) *Město*$_{\text{ACT}}$ *získalo od vlády*$_{\text{ORIG}}$ *dotaci*$_{\text{CPHR}}$ *7 milionů*$_{\text{PAT}}$ *korun.*
'The town$_{\text{ACT}}$ obtained a grant$_{\text{CPHR}}$ of 7 million$_{\text{PAT}}$ crowns from the government$_{\text{ORIG}}$.'

3.1.4. Coreference and its Key Role in LVCs

As already mentioned in Section 3.1.2, in LVCs semantically underspecified valency complementations of light verbs obtain their semantic specifications – the principal role in this process is played by coreference between valency complementations of the predicative noun and the light verb.

The most prominent coreferential relation is the one between ACT of a predicative noun with a certain complementation of the light verb. Its prominence is also empirically attested by the corpus material provided by the Prague Dependency Treebank (PDT, Bejček et al., 2013):[7] from 1,695 LVCs with predicative nouns expressed as prepositionless accusative objects of light verbs in PDT, 1,609 LVCs (95% in total) are characterized by the coreference of the nominal ACT and a certain verbal complementation; the remaining 5% represents rather annotation errors (see Kettnerová and Bejček, 2016). The presence of a pair of ACT of the predicative noun and a certain valency complementation of the light verb referring to the same extralinguistic entity can be thus adopted as a definitional criterion for delimiting LVCs. See the test of coreference in Radimský (2010) as well.

Semantically underspecified valency complementations of a light verb can enter into different coreferential relations, depending on valency structure of the predicative nouns selecting the given verb. For example, in the valency frame of the light verb *přinést* 'to bring' in its non-causative function, the ACT and ADDR are semantically unsaturated, see the valency frame of this verb in (20). Three types of coreference of this ACT and ADDR with complementations of predicative nouns are attested in the corpus data, as shown below by simplified dependency trees. Additional two types of coreference appear for the given verb with causative function, when the 'Causator' is mapped onto the verbal ACT. Table 1 below provides more examples of LVCs with the given verb for all the mentioned types of coreference.

I. ACT$_{\text{V}}$ – ACT$_{\text{N}}$ & ADDR$_{\text{V}}$ – ADDR$_{\text{N}}$
First, the ACT of the light verb *přinést* 'to bring' corefers with the nominal ACT and the

[7] http://ufal.mff.cuni.cz/pdt3.0/ (http://hdl.handle.net/11858/00-097C-0000-0023-1AAF-3)

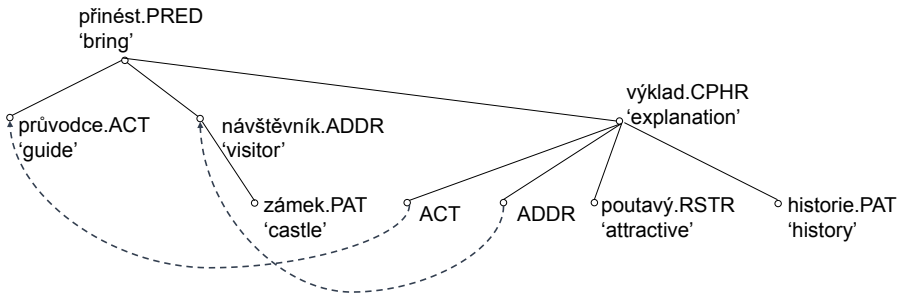


Figure 1. The simplified dependency tree of the LVC 'přinést výklad' 'to bring an explanation' in sentence (24) (the dashed arrows link coreferring valency complementations).

ADDR of the verb corefers with the nominal ADDR. This type of coreference is characteristic, e.g., of the LVC *přinést výklad* 'to bring an explanation'. See the simplified dependency tree of its example sentence (24) in Figure 1. The valency frames of the light verb (20) and of the predicative noun (21) and the scheme of the mapping of semantic participants in the given LVC (23) are provided below.

(20) *přinést* 'to bring': $\mathsf{ACT}_{\mathsf{nom}}\ \mathsf{ADDR}_{\mathsf{dat}}\ \mathsf{CPHR}_{\mathsf{acc}}$

(21) *výklad* 'explanation': $\mathsf{ACT}_{\mathsf{gen,pos,instr,od+gen}}\ \mathsf{ADDR}_{\mathsf{dat}}\ \mathsf{PAT}_{\mathsf{gen,pos,o+loc,dcc}}$

(22) *průvodcův*$_{\mathsf{ACT.poss}}$ *poutavý výklad historie*$_{\mathsf{PAT.gen}}$ *návštěvníkům*$_{\mathsf{ADDR.dat}}$ *zámku*
'guide's$_{\mathsf{ACT}}$ attractive explanation of history$_{\mathsf{PAT}}$ to visitors$_{\mathsf{ADDR}}$ of the castle'

(23) *přinést výklad* 'to bring an explanation':

$$\begin{array}{lcllll} & & \mathsf{ACT_V} & \leftrightarrow & \mathsf{ACT_N} & \Leftrightarrow \text{ 'Speaker'} \\ & & \mathsf{ADDR_V} & \leftrightarrow & \mathsf{ADDR_N} & \Leftrightarrow \text{ 'Recipient'} \\ \textit{výklad} \text{ 'explanation'} & \equiv & \mathsf{CPHR_V} & & & \\ & & & & \mathsf{PAT_N} & \Leftrightarrow \text{ 'Information'} \end{array}$$

(24) *Průvodce*$_{\mathsf{ACT.nom}}$ *přinesl návštěvníkům*$_{\mathsf{ADDR.dat}}$ *zámku poutavý výklad*$_{\mathsf{CPHR.acc}}$ *historie*$_{\mathsf{PAT.gen}}$.
'The guide$_{\mathsf{ACT}}$ brought an attractive explanation$_{\mathsf{CPHR}}$ of history to visitors$_{\mathsf{ADDR}}$ of the castle.'

II. $\mathsf{ACT_V}$ – $\mathsf{ACT_N}$ & $\mathsf{ADDR_V}$ – $\mathsf{PAT_N}$

The second type represents an infrequent type,[8] characterizing, e.g., the LVC *přinést efekt* 'to bring an effect', see the dependency tree in Figure 2 of its example sentence (28).

[8] PAT with predicative nouns in these cases often corresponds to cognitively shifted ADDR.

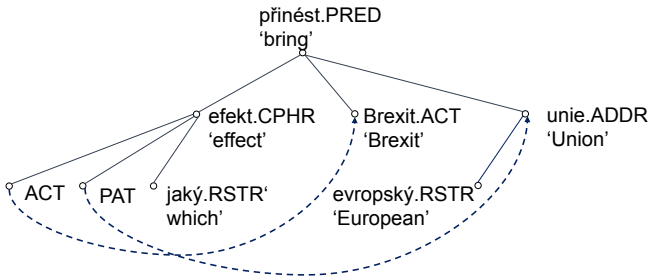


Figure 2. The simplified dependency tree of the LVC 'přinést efekt' 'to bring an effect' in sentence (28).

(25) *efekt* 'effect': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{na+acc,pro+acc}}$

(26) *efekt Brexitu*$_{\mathsf{ACT.gen}}$ *na Evropskou unii*$_{\mathsf{PAT.na+acc}}$
'an effect of Brexit$_{\mathsf{ACT}}$ on the European Union$_{\mathsf{PAT}}$'

(27) *přinést efekt* 'to bring an effect':

$$\begin{array}{lcl} \mathsf{ACT_V} & \leftrightarrow & \mathsf{ACT_N} \quad \Leftrightarrow \quad \text{'Influencer'} \\ \mathsf{ADDR_V} & \leftrightarrow & \mathsf{PAT_N} \quad \Leftrightarrow \quad \text{'Influencee'} \end{array}$$

efekt 'effect' $\equiv$ $\mathsf{CPHR_V}$

(28) *Jaký efekt*$_{\mathsf{CPHR.acc}}$ *přinese Brexit*$_{\mathsf{ACT.nom}}$ *Evropské unii*$_{\mathsf{ADDR.dat}}$*?*
'Which effect$_{\mathsf{CPHR}}$ brings Brexit$_{\mathsf{ACT}}$ to the European Union$_{\mathsf{ADDR}}$?'

III. $\mathbf{ACT_V - PAT_N}$ & $\mathbf{ADDR_V - ACT_N}$

The third type of coreference is typical of, e.g., the LVC *přinést zklamání* 'to bring disappointment', see the dependency tree in Figure 3 of example sentence (32).

(29) *zklamání* 'disappointment': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{nad+instr,z+gen,inf,dcc}}$

(30) *zklamání občanů*$_{\mathsf{ACT.gen}}$ *z výsledků*$_{\mathsf{PAT.z+gen}}$ *voleb*
'disappointment of citizens$_{\mathsf{ACT}}$ by the results$_{\mathsf{PAT}}$ of the elections'

(31) *přinést zklamání* 'to bring disappointment':

$$\begin{array}{lcl} \mathsf{ACT_V} & \leftrightarrow & \mathsf{PAT_N} \quad \Leftrightarrow \quad \text{'Stimulus'} \\ \mathsf{ADDR_V} & \leftrightarrow & \mathsf{ACT_N} \quad \Leftrightarrow \quad \text{'Experiencer'} \end{array}$$

zklamání 'disappointment' $\equiv$ $\mathsf{CPHR_V}$

(32) *Výsledky*$_{\mathsf{ACT.nom}}$ *voleb přinesly občanům*$_{\mathsf{ADDR.dat}}$ *zklamání*$_{\mathsf{CPHR.acc}}$.
'The results$_{\mathsf{ACT}}$ of the elections brought disappointment$_{\mathsf{CPHR}}$ to citizens$_{\mathsf{ADDR}}$.'

Besides its non-causative function, the light verb *přinést* 'to bring' can serve as the causative verb as well (see Section 3.1.2). The 'Causator' provided by this light verb is

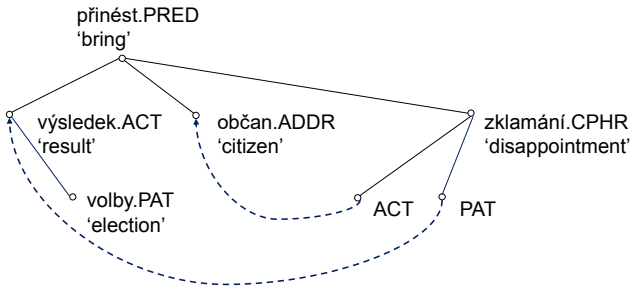


Figure 3. The simplified dependency tree of the LVC 'přinést zklamání' 'to bring disappointment' in sentence (32).

mapped onto its ACT. Thus the only remaining semantically underspecified complementation in the valency frame of the given verb is the ADDR, see the valency frame in (20). LVCs with the causative light verb *přinést* 'to bring' are then characterized by two types of coreference.

IV. 'Causator': $\mathbf{ACT_V}$ & $\mathbf{ADDR_V}$ – $\mathbf{ACT_N}$
This type is characteristic of, e.g., the LVC *přinést poznatek* 'to bring knowledge', see the dependency tree of example sentence (36) in Figure 4.

(33) *poznatek* 'knowledge': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{gen,k+dat,o+loc,dcc}}$ $\mathsf{ORIG}_{\mathsf{z+gen}}$

(34) *naše*$_{\mathsf{ACT.gen}}$ *poznatky o vodě*$_{\mathsf{PAT.o+loc}}$ *na Marsu*
'our$_{\mathsf{ACT}}$ knowledge about water$_{\mathsf{PAT}}$ on Mars'

(35) *přinést poznatek* 'to bring knowledge':

'Causator'	$\Leftrightarrow$	$\mathsf{ACT_V}$				
		$\mathsf{ADDR_V}$	$\leftrightarrow$	$\mathsf{ACT_N}$	$\Leftrightarrow$	'Cognizer'
poznatek 'knowledge'	$\equiv$	$\mathsf{CPHR_V}$				
				$\mathsf{PAT_N}$	$\Leftrightarrow$	'Content'

(36) *Sonda*$_{\mathsf{ACT.nom}}$ *nám*$_{\mathsf{ADDR.dat}}$ *přinesla poznatky*$_{\mathsf{CPHR.acc}}$ *o vodě*$_{\mathsf{PAT.o+loc}}$ *na Marsu.*
'The space probe$_{\mathsf{ACT}}$ brought us$_{\mathsf{ADDR}}$ knowledge$_{\mathsf{CPHR}}$ about water$_{\mathsf{PAT}}$ on Mars.'

V. 'Causator': $\mathbf{ACT_V}$ & $\mathbf{ADDR_V}$ – $\mathbf{PAT_N}$ & $\mathbf{LOC_V}$ – $\mathbf{ACT_N}$
This coreference characterizes, e.g., the LVC *přinést přízeň* 'to bring favor', see the dependency tree in Figure 5 representing example sentence (40).

(37) *přízeň* 'favor': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{dat,k+dat,pro+acc,vůči+dat}}$

(38) *přízeň publika*$_{\mathsf{ACT.gen}}$ *ke zpěvákovi*$_{\mathsf{PAT.k+dat}}$
favor of audience$_{\mathsf{ACT}}$ to the singer$_{\mathsf{PAT}}$

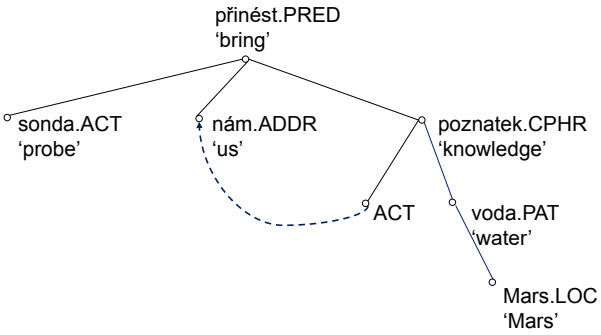


Figure 4. The simplified dependency tree of the LVC ‘přinést poznatek’ ‘to bring knowledge’ in sentence (36).

(39) *přinést přízeň* ‘to bring favor’:

‘Causator’	⇔	ACT_V				
		$ADDR_V$	↔	PAT_N	⇔	‘Stimul’
		LOC_V	↔	ACT_N	⇔	‘Experiencer’
přízeň ‘favor’	≡	$CPHR_V$				

(40) *Zdařilé turné*$_{\text{ACT.nom}}$ *přineslo zpěvákovi*$_{\text{ADDR.dat}}$ *přízeň*$_{\text{CPHR.acc}}$ *u publika*$_{\text{LOC.u+gen}}$.
‘The successful tour$_{\text{ACT}}$ brought the singer$_{\text{ADDR}}$ favor$_{\text{CPHR}}$ of an audience.’

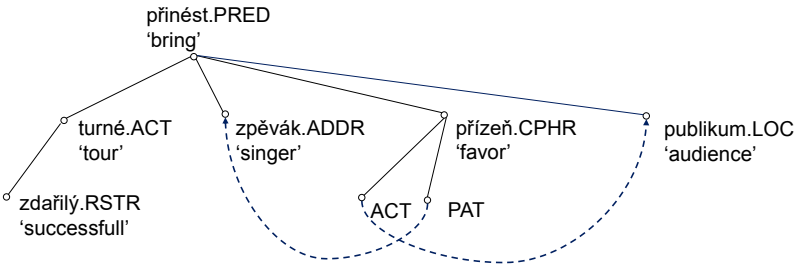


Figure 5. The simplified dependency tree of the LVC ‘přinést přízeň’ ‘to bring favor’ in sentence (40).

Ambiguous mappings

Several light verbs can function ambiguously in a single LVC with respect to non-causative and causative function. For example, the light verb *přinést* ‘to bring’ in the LVC *přinést zklamání* ‘to bring disappointment’ can serve either as non-causative, see above

type III (mapping (31) and example (32)), or as causative one. The latter case belongs to type IV as it exhibits the same coreference as e.g the LVC *přinést poznatek* 'to bring knowledge', see the coreference in (41) and example (42).

(41) *přinést zklamání* 'to bring disappointment':

'Causator'	$\Leftrightarrow$	ACT_V				
		$ADDR_V$	$\leftrightarrow$	ACT_N	$\Leftrightarrow$	'Experiencer'
				PAT_N	$\Leftrightarrow$	'Stimulus'
zklamání 'disappointment'	$\equiv$	$CPHR_V$				

(42) *Radikální dieta*$_{ACT.nom}$ *Janě*$_{ADDR.dat}$ *přinesla jen další zklamání*$_{CPHR.acc}$ *z vlastního selhání*$_{PAT.z+gen}$.
'Radical diet$_{ACT}$ brought Jane$_{ADDR}$ disappointment$_{CPHR}$ by her own failure$_{PAT}$.'

---

type I. $ACT_V - ACT_N$ & $ADDR_V - ADDR_N$
přinést 'to bring': *důkaz* 'evidence', *informace* 'information', *nabídku* 'offer', *návod* 'instruction', *odpověď* 'answer', *podnět* 'impulse', *ponaučení* 'lesson', *výklad* 'explanation', *zprávu* 'message';

type II. $ACT_V - ACT_N$ & $ADDR_V - PAT_N$
přinést 'to bring': *efekt* 'effect';

type III. $ACT_V - PAT_N$ & $ADDR_V - ACT_N$
přinést 'to bring': *neklid* 'uneasiness', *potěšení* 'pleasure', *potíž* 'trouble', *problém* 'trouble', *prospěch* 'benefit', *překvapení* 'surprise', *radost* 'to bring happiness', *uspokojení* 'satisfaction', *užitek* 'benefit', *úleva* 'relief', *útěcha* 'comfort', *zklamání* 'disappointment';

type IV. Causator: ACT_V & $ADDR_V - ACT_N$
přinést 'to bring': *mír* 'calm', *možnost* 'opportunity', *neklid* 'uneasiness', *neštěstí* 'bad luck', *obživa* 'living', *omezení* 'restriction', *pokrok* 'progress', *popularita* 'popularity', *potěšení* 'pleasure', textitpotíž 'trouble', *poznatek* 'insight', *problém* 'trouble', *prospěch* 'benefit', *přátelství* 'friendship', *překvapení* 'surprise', *radost* 'happiness', *riziko* 'risk', *smůla* 'bad luck', *štěstí* 'happiness', *uspokojení* 'satisfaction', *uznání* 'recognition', *užitek* 'benefit', *úleva* 'relief', *úspěch* 'success', *úspora* 'savings', *útěcha* 'comfort', *výhoda* 'advantage', *zisk* 'profit', *zjištění* 'findings', *zklamání* 'disappointment', *zkušenost* 'experience';

type V. Causator: ACT_V & $ADDR_V - PAT_N$ & $LOC_V - ACT_N$
přinést 'to bring': *přízeň* 'favor'.

Table 1. Examples of LVCs with the verb 'přinést' 'to bring' for all relevant types of coreference.

3.2. Surface Syntactic Structure

The surface structure formation of LVCs shares the same basic principle as surface constructions with full verbs according to which each semantic participant is expressed on the surface just once.

When a semantic participant in LVCs corresponds either to a valency complementation of the predicative noun, or to a complementation of the light verb (in case of causative light verbs), the situation is obvious: the surface syntactic expression of such semantic participant can be stipulated only by morphemic forms of the valency complementation (be it nominal, or verbal) to which the given participant corresponds.

However, when a semantic participant is mapped onto a valency complementation of the predicative noun and at the same time (via coreference) onto a complementation of the light verb, it raises a question which of the coreferring valency complementations is expressed on the surface. In these cases, morphemic forms prescribed by individual valency complementations serve as an important clue for the identification of the valency complementations expressed on the surface. As the analysis of extensive corpus data corroborates, these semantic participants are mostly expressed on the surface as valency complementations of the light verb.

For example, let us analyze the surface structure of the LVC *uložit úkol* 'to give a task' in (47): from the semantic participants characterizing this LVC, 'Speaker' and 'Recipient' are mapped onto the ACT and ADDR of the predicative noun and at the same time via coreference onto the ACT and ADDR of the light verb as well, respectively (the scheme of the mapping (46)). 'Speaker' is expressed in the surface structure of the given LVC as the subject corresponding to the verbal ACT as its morphemic form (the nominative case) unequivocally determines, see ACT in the valency frame of the light verb in (43) and ACT in the frame of the noun in (44).

As for 'Recipient', its dative case does not unequivocally indicate the surface position: as both the verbal ADDR and the nominal ADDR can have the form of the dative case, see the valency frames of the light verb (43) and the predicative noun (44), it can be either an indirect object with the function of the verbal ADDR, or an attribute with the function of the nominal ADDR. Thus a question arises which of these complementations the given dative case expresses. The surface syntactic behavior of this complementation in diatheses indicates that it can be accounted for as the verbal ADDR: verbal complementations – in contrast to nominal ones – are sensitive to surface syntactic shifts in diatheses. When recipient passive diathesis is applied to the LVC *uložit úkol* 'to give a task', the given ADDR changes its form from the dative into the nominative, see (48), which clearly manifests that it is the ADDR governed by the light verb. Moreover, the possibility of word order changes (compare (47) and (49)) supports its analysis as verbal ADDR as well.

'Obligation', mapped just onto the PAT of the predicative noun, can be expressed only as the attribute as morphemic forms of the given complementation require, see the valency frame of the noun (44).

(43) *uložit* 'to give': $\mathsf{ACT}_{\mathsf{nom}}\ \mathsf{ADDR}_{\mathsf{dat}}\ \mathsf{CPHR}_{\mathsf{acc}}$

(44) *úkol* 'task': $\mathsf{ACT}_{\mathsf{gen,pos,od+gen}}\ \mathsf{ADDR}_{\mathsf{gen,dat,pos,pro+acc}}\ \mathsf{PAT}_{\mathsf{inf,dcc}}$

(45) *učitelův*$_{\mathsf{ACT.pos}}$ *úkol žákům*$_{\mathsf{ADDR.dat}}$ *narýsovat*$_{\mathsf{PAT.inf}}$ *krychli*
'teacher's$_{\mathsf{ACT}}$ task to students$_{\mathsf{ADDR}}$ to draw$_{\mathsf{PAT}}$ a cube'

(46) *uložit úkol* 'to give a task':

		$\mathsf{ACT}_{\mathsf{V}}$	$\leftrightarrow$	$\mathsf{ACT}_{\mathsf{N}}$	$\Leftrightarrow$	'Speaker'
		$\mathsf{ADDR}_{\mathsf{V}}$	$\leftrightarrow$	$\mathsf{ADDR}_{\mathsf{N}}$	$\Leftrightarrow$	'Recipient'
úkol 'task'	$\equiv$	$\mathsf{CPHR}_{\mathsf{V}}$				
				$\mathsf{PAT}_{\mathsf{N}}$	$\Leftrightarrow$	'Obligation'

(47) *Učitel*$_{\mathsf{ACT.subj.nom}}$ *uložil žákům*$_{\mathsf{ADDR.inobj.dat}}$ *úkol*$_{\mathsf{CPHR.obj.acc}}$ *narýsovat*$_{\mathsf{PAT.attr.inf}}$ *krychli.*
'The teacher$_{\mathsf{ACT}}$ gave students$_{\mathsf{ADDR}}$ a task$_{\mathsf{CPHR}}$ to draw$_{\mathsf{PAT}}$ a cube.'

(48) *Žáci*$_{\mathsf{ADDR.subj.nom}}$ *dostali od učitele*$_{\mathsf{ACT.inobj.od+gen}}$ *uložen úkol*$_{\mathsf{CPHR.obj.acc}}$ *narýsovat*$_{\mathsf{PAT.attr.inf}}$ *krychli.*
'The students$_{\mathsf{ADDR}}$ were assigned a task$_{\mathsf{CPHR}}$ by the teacher$_{\mathsf{ACT}}$ to draw$_{\mathsf{PAT}}$ a cube.'

(49) *Žákům*$_{\mathsf{ADDR.inobj.dat}}$ *učitel*$_{\mathsf{ACT.subj.nom}}$ *uložil úkol*$_{\mathsf{CPHR.obj.acc}}$ *narýsovat*$_{\mathsf{PAT.attr.gen}}$ *krychli.*

3.2.1. Double Expression of a Semantic Participant

There are basically two exceptions from general principles underlying the surface realization of semantic participants. First, in rare cases a semantic participant mapped onto the ACT of a predicative noun and at the same time onto the ACT of a light verb can be expressed twice on the surface. For example, in the LVC *přinést výklad* 'to bring an explanation' in (50), the 'Speaker' is expressed twice in the surface structure, as the subject with the function of the verbal ACT and at the same time as an attribute with the function of the nominal ACT.

(50) *Průvodce*$_{\mathsf{ACT.subj.nom}}$ *přinesl návštěvníkům*$_{\mathsf{ADDR.inobj.dat}}$ *zámku svůj*$_{\mathsf{ACT.attr.pos}}$ *vlastní výklad*$_{\mathsf{CPHR.obj.acc}}$ *historie*$_{\mathsf{PAT.attr.gen}}$.
'The guide$_{\mathsf{ACT}}$ brought his$_{\mathsf{ACT}}$ own explanation$_{\mathsf{CPHR}}$ of history to visitors$_{\mathsf{ADDR}}$ of the castle.'

3.2.2. Semantic Participants Mapped onto an Optional Free Modification of the Light Verb

Second, in those cases in which a semantic participant is mapped onto an actant of the predicative noun and via coreference onto an optional free modification of the light verb, the given semantic participant can be realized either as the nominal valency complementation, or as the verbal one.

For example, in the LVC *probouzet vzpomínku* 'to raise memory', three semantic participants can be expressed on the surface: 'Causator' contributed to the LVC by the causative light verb *probouzet* 'to raise' and two semantic participants – 'Cognizer'

and 'Experience' – provided by the predicative noun *vzpomínka* 'memory', see the valency frame of the given light verb in (51) and the frame of the noun in (52). The LVC *probouzet vzpomínku* 'to raise memory' is characterized by the coreference of the ACT of the predicative noun with the optional LOC of the light verb, see the scheme of the mapping of participants in (54). As a result, this semantic participant can be expressed on the surface either as the verbal LOC (realized on the surface as a local adverbial), see example (55), or as the nominal ACT (expressed on the surface as an attribute), see example (56).

(51) *probouzet* 'to raise': $\mathsf{ACT}_{\mathsf{nom,inf,dcc}}$ $\mathsf{CPHR}_{\mathsf{acc}}$ LOC

(52) *vzpomínka* 'memory': $\mathsf{ACT}_{\mathsf{gen,pos}}$ $\mathsf{PAT}_{\mathsf{na+acc,dcc}}$

(53) $moje_{\mathsf{ACT.pos}}$ $vzpomínka\ na\ mládí_{\mathsf{PAT.na+acc}}$
'my_{ACT} memory of (my) $\text{youth}_{\mathsf{PAT}}$'

(54) *probouzet vzpomínku* 'to raise memory':

'Causator'	⇔	$\mathsf{ACT}_{\mathsf{V}}$				
vzpomínka 'memory'	≡	$\mathsf{CPHR}_{\mathsf{V}}$				
		$\mathsf{LOC}_{\mathsf{V}}$	↔	$\mathsf{ACT}_{\mathsf{N}}$	⇔	'Cognizer'
				$\mathsf{PAT}_{\mathsf{N}}$	⇔	'Experience'

(55) $Hudba_{\mathsf{ACT.subj.nom}}$ $ve\ mně_{\mathsf{LOC.adv.v+loc}}$ $probouzela\ vzpomínky_{\mathsf{CPHR.obj.acc}}$ $na\ mládí_{\mathsf{PAT.attr.na+acc}}$.
$\text{music}_{\mathsf{ACT}}$ in me_{LOC} raised $\text{memories}_{\mathsf{CPHR}}$ on $\text{youth}_{\mathsf{PAT}}$

(56) $Hudba_{\mathsf{ACT.subj.nom}}$ $probouzela\ moje_{\mathsf{ACT.attr.pos}}$ $vzpomínky_{\mathsf{CPHR.obj.acc}}$ $na\ mládí_{\mathsf{PAT.attr.na+acc}}$.
$\text{music}_{\mathsf{ACT}}$ raised my_{ACT} $\text{memories}_{\mathsf{CPHR}}$ on $\text{youth}_{\mathsf{PAT}}$
'The music raised my memories of my youth.'

3.2.3. Principles Governing the Expression of Semantic Participants

The principles governing the expression of semantic participants in the surface structure of LVCs can be summarized as follows:[9]

- A semantic participant mapped onto a single valency complementation is expressed in a surface syntactic position:
  - of the respective valency complementation of the predicative noun (be it actant, or free modification),
  - of the respective valency complementation of the light verb (representing 'Causator').
- A semantic participant mapped onto two coreferring valency complementations is expressed in a surface syntactic position:
  - either of the respective valency complementation of the light verb (be it actant, or free modification),

[9] As the double expression of a semantic participant in LVCs discussed above in Section 3.2.1 is highly restricted, it is not reflected in the given principles.

– or of the respective valency complementation of the predicative noun if the coreferring verbal complementation is an optional free modification.

4. Formal Model of Lexicographic Representation of LVCs

The lexicographic representation of LVCs requires a close cooperation of both the lexical and the grammar part of the language description. In this section, we describe how the information on LVCs is reflected in the data and grammar component of the VALLEX lexicon, see above Section 2. As we have demonstrated above, the deep and surface syntactic structure of LVCs can be inferred from valency structures of both light verbs and predicative nouns and coreferential relations between individual valency complementations on the rule basis. These rules, described in the grammar component of the lexicon, operate on the information provided by its data component.

4.1. Data Component

In the data component, three special attributes have been introduced providing the information necessary for deriving deep and surface syntactic structure of LVCs.

Attribute `lvc`. The data component of the VALLEX lexicon stores lexical units of both predicative nouns and light verbs,[10] providing their core valency characteristics in the form of valency frames, as described in Section 3.1.1 and 3.1.2. The respective lexical units are interlinked via the special attribute `lvc`: with each lexical unit of a predicative noun, this attribute provides references to lexical units of the light verbs selected by the given noun, whereas with each lexical unit of a light verb, links to the predicative nouns with which the light verb combines are provided.

Attribute `map`. This attribute is provided within lexical units of light verbs. It stores the information on coreference between valency complementations of light verbs and complementations of predicative nouns – the coreference is represented as a set of pair(s) of coreferring valency complementations, the valency complementations being represented by their respective functors (and for user's convenience indicated in the lower index either as `V`, or as `N`, distinguishing verbal complementations from nominal ones).

Attribute `instig`. This attribute is provided within lexical units of causative light verbs – it gives the valency complementation from the respective valency frames onto

[10] Although collocations of predicative nouns with light verbs form multiword lexical units, we use here the term lexical units to refer to individual predicative nouns and light verbs, stressing their compositional possibilities. We are aware that in case of light verbs, this term is not, strictly speaking, correct: light verbs are not able to occur outside the collocation with predicative nouns (giving evidence of their semi-lexical status). However, from the formal point of view, light verbs can be treated similarly as individual senses of full verbs.

which the semantic participant 'Causator' is mapped, the given complementation being represented by its respective functor.

Each valency frame of a light verb is assigned with a set of pair(s) or triplet(s) (if the attribute `instig` is relevant) of the above given attributes, distinguishing different coreference relations and eventually causative function of the light verb; relevant pairs (or triplets) are differentiated by Arabic numerals. In addition, each set is accompanied with examples illustrating individual LVCs. See the lexical entry of the light verb *uložit* 'to impose' in Figure 8 and the entry of the predicative noun *úkol* 'task' in Figure 7.

4.2. Grammar Component

In the grammar component, formal rules governing both the deep and surface structure formation of LVCs are stored, operating on the information provided by the data component, as follows:

The *deep syntactic structure of LVCs* consists of:

(i) all valency complementations of the light verb provided by its valency frame (attribute `frame`),
(ii) all valency complementations of the predicative noun provided by its valency frame (attribute `frame`),
(iii) coreferential relations between individual valency complementations of the light verb and complementations of the predicative noun (attribute `map`).

The *surface syntactic structure of LVCs* comprises:[11]

A. syntactic positions of all valency complementations of the light verb, namely:
   (i) the syntactic position of the predicative noun
   (its morphemic form is given by the `CPHR` valency complementation),
   (ii) the syntactic position of 'Causator', if relevant
   (its morphemic form is given by the valency complementation provided in the attribute `instig`),
   (iii) the syntactic position(s) of other valency complementations which corefer with complementations of the predicative noun (only optional free modifications may remain unexpressed)
   (morphemic forms of these positions are given by the respective valency complementations provided in the attribute `map`).
B. syntactic positions of those valency complementations of the predicative noun that satisfy the following conditions:

[11] These principles summarize the surface structure formation of LVCs on which further surface syntactic operations leading to deletion of valency complementations can be applied due to their optionality, actual ellipsis, generalization etc.

(iv) the syntactic position(s) of the valency complementations that do not corefer with any complementations of the light verb
(morphemic forms of these positions are given by the respective valency complementations of the predicative noun),
(v) the syntactic position(s) of the valency complementations that corefer with optional free modifications of the light verbs not expressed on the surface
(morphemic forms of these positions are given by the respective valency complementation of the predicative noun).[12]

5. Annotation of LVCs in VALLEX

5.1. Lexical Stock

Each LVC is formed by a collocation consisting of a predicative noun and a light verb. The large amount of such collocations in Czech is not easily manageable, thus, some selection criteria had to be determined at the beginning of the annotation process. For the identification of the collocations representing LVCs, we first had to identify an inventory of verb lemmas already stored in the VALLEX lexicon that can function as light verbs. For this purpose, we used the valency lexicon PDT-Vallex (Urešová, 2011). In this lexicon, valency frames of light verbs are indicated by the functor CPHR, labeling the valency position of predicative nouns. On this basis, we automatically identified 124 verb lemmas that have at least one valency frame in PDT-Vallex with the CPHR functor. The intersection of verb lemmas obtained from PDT-Vallex and of verb lemmas contained in VALLEX was 105 in total. As VALLEX treats aspectual counterparts of verbs as a single verb lexeme, respective aspectual counterparts have been added (if not already in the list). The resulting number of 145 verb lemmas has formed the inventory of verbs selected for the further annotation.

To identify the most frequent and most salient predicative nouns which select the given light verbs, we used the Sketch Engine (Kilgarriff et al., 2014), a corpus tool allowing users to obtain summaries of words' grammatical and collocational properties. A balanced corpus of synchronous texts SYN2010[13] was used as the material base. In the first step, the collocation lists of all selected verb lemmas were obtained. From these lists, only the nominal collocates expressed as direct object in the accusative case (representing the central and most frequent cases of light verb collocations in Czech) were selected: almost 3,050 noun lemmas (21 nouns on average for a verb lemma); see also Kettnerová et al. (2016).

[12] In rare cases, the syntactic position of the nominal ACT coreferring with the verbal ACT is expressed in the surface structure (typically as a possessive pronoun), in addition to its expression in the verbal position according to principle (Aiii), compare Section 3.2.1 and example (50).

[13] Czech National Corpus – SYN2010. Institute of the Czech National Corpus, Faculty of Arts, Charles University, Prague 2010. Available from http://www.korpus.cz.

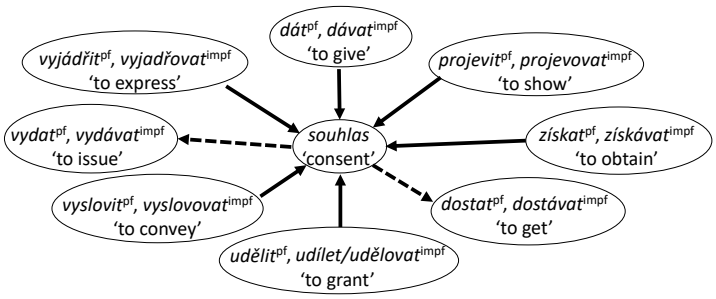


Figure 6. Collocations of the predicative noun 'souhlas' 'consent' with light verbs, found with the Sketch Engine; the collocations marked by the continuous arrow have been identified in the lists of nominal collocates of the given light verbs 'dát, dávat' 'to give', 'projevit, projevovat' 'to show' etc.; the collocation joined by the dashed arrow have been extracted from the list of verbal collocates of the noun 'souhlas' 'consent').

A human annotator has been asked to indicate only those nouns in each list that represent predicative nouns forming collocations with the given light verb. As the main criterion for distinguishing collocates with light verbs from those with full verbs, the coreference of the nominal ACT with a valency complementation of the light verb has been adopted (see Section 3.1.4). The collocations that satisfy this condition are interpreted as LVCs.

In the second step, salient collocations of the predicative nouns have been added as well, using the Sketch Engine: for each of the predicative noun obtained in the first step, its missing relevant collocations with light verbs have been automatically extracted from its verb collocation list (in this step, we restricted the verbs to the list of 145 already identified light verb lemmas). See Figure 6, displaying collocations of the noun *souhlas* 'consent' obtained in the first and second steps of the annotation process.

The resulting number of collocations from the first and second step sums up to 2,991 collocations in total (counted as combinations of a lemma of a light verb and a lemma of a predicative noun). These collocations represent the lexical stock integrated into the VALLEX lexicon.

5.2. Annotation Process

In the next step, the selected 2,991 collocations of predicative nouns with light verbs have been assigned with the relevant information, as introduced in Section 4. Each verb lemma and noun lemma have been processed separately, and then interlinked into a relevant LVC.

5.2.1. Annotation of Predicative Nouns

As the VALLEX lexicon originally contained the information only on valency behavior of verbs, it was necessary to add the same information on the selected predicative nouns. The data format of the lexicon, as described in Section 2, is suitable also for the description of other parts-of-speech. We thus made use of the same structure of lexical entries as designed for verbs (excluding verb-specific optional attributes).

The lexical entry of a predicative noun comprises its noun lexeme, i.e., a two-fold unit associating all forms of the given noun with its lexical units (its individual senses). Each lexeme is represented by the respective noun lemma of the predicative noun, or (if relevant) by more than one lemma.

úkol

1 ≈ zadání; skutečnost určená k vykonání `assignment'
-frame: **ACT$_{2,pos,od+2}$ ADDR$_{2,3,pos,pro+4}$ PAT$_{inf,dcc}$**
-lu: úkol-1
-lvc: **ukládat-uložit-8**, zadat-zadávat-4, klást-4, přebírat-přebrat-převzít-2.1, uvalit-uvalovat-2
-example: Pracovní úkol zaměstnavatele.ACT vyplývá zejména z popisu práce …;
V tomto směru jsou úkoly od vedení.ACT klubu celkem jasné.;
Hráči přesně plní trenérovy.ACT úkoly.;
Hosté prohrávali a jejich.ADDR úkolem od trenéra.ACT bylo neprohrát utkání o deset branek.;
domácí úkoly pro předškoláky.ADDR;
těžký úkol vybrat.PAT vítěze.;
Výsledkem byl úkol Komisi.ADDR, aby začala.PAT připravovat pro tři země akční programy.;
Nový čínský úkol zní: ať naši zpěváci dobudou.PAT svět.;
Z rady vyvstal úkol, že bychom měli svolat.PAT obecnou diskuzi.
-control: ADDR

2 ≈ poslání; úloha; funkce `mission; role; function'
-frame: **ACT$_{2,pos}$ PAT$_{inf,dcc}$**
-lu: úkol-2
-lvc: vykonat-vykonávat-2, přebírat-přebrat-převzít-2.2
-example: Úkolem řešitele.ACT sudoku je v co nejkratším čase doplnit prázdná místa v tabulce.;
Úkolem závodníka.ACT bude předvést odvážný improvizovaný skok do vody.;
Jaký je náš.ACT úkol v dějinách člověčenstva?;
Před zápasem byl můj.ACT úkol určit.PAT brankářku, která nastoupí do zápasu.;
Je to jeho.ACT úkol, aby zakázka byla.PAT čistá a hotová včas.;
To je taky jeho.ACT hlavní úkol, ať sešívky zapomenou.PAT na poháry a boje o místo nahoře.;
Jeho.ACT předpokládaný úkol, že má sedět.PAT při každém přelíčení a podat.PAT čtenářům jeho průběh, byl nesmírně obtížný.
-control: ACT

Figure 7. Lexical entry of the predicative noun 'úkol' 'task' (two lexical units).

The key information on the valency behavior of individual lexical units of the predicative noun is provided by valency frames, exemplified by illustrative examples. Morphemic forms in valency frames of the noun describe the usage of its lexical units in nominal structures, see Section 3.1.1.

For the purpose of the description of LVCs, each lexical unit of a predicative noun is assigned with the special attribute `lvc` providing a list of references to individual light verbs with which the given noun forms LVCs. See the illustrative example of the lexical entry of the predicative noun *úkol* 'task' in Figure 7.

We have restricted the number of syntactically annotated predicative nouns to those that form LVCs in the data with at least two light verbs (277 noun lexemes represented by 284 noun lemmas). Those nouns that represent a part of LVCs with one light verb are provided only in the form of a list of noun lemmas in the attribute `lvc` in the lexical entries of the respective light verbs (if these noun lemmas are counted, the number of predicative nouns increases to 577 lemmas). The basic statistics on annotated predicative nouns is provided in Table 2.

	Lemmas	Lexemes	Lexical Units
Predicative nouns	284 (577)	277	350
Light verbs	145	78	117

Table 2. The basic statistics on the annotated predicative nouns and the light verbs in VALLEX. The number of noun lemmas in parenthesis includes also predicative nouns which form LVCs with only one light verb.

5.2.2. Annotation of Light Verbs

In the lexical entry of each verb lemma indicated in the previous step as a lemma representing a light verb, a relevant lexical unit (or more lexical units) of the given light verb has been identified. If no relevant lexical unit has been comprised in the lexical entry, it has been manually added. Then each identified lexical unit has been subject to necessary adjustment.

First, in each valency frame of relevant lexical units, a valency complementation standing for predicative nouns has been labeled with the CPHR functor. Second, each lexical unit of the light verb has been ascribed with one or more sets of the following attributes (as introduced in Section 4): (i) the attribute `lvc`, providing references to lexical units of the predicative nouns with which the given light verb form LVCs, and (ii) the attribute `map`, introducing the information on coreference characterizing the given LVCs. (iii) In case of causative light verbs, each set is further supplemented with the attribute `instig`, storing the information on the valency position of 'Causator'. If more

ukládat^{impf} , uložit^{pf}

....

8 ≈ light verb `to give; to assign' (as light verb)
-frame: **ACT$_1$ ADDR$_3$ CPHR$_4$**
-lu: ukládat-uložit-8
-lvc1: **úkol-1**, zákaz-1
-map1: ACTv-ACTn, ADDRv-ADDRn
-example1:
impf: Prostřednictvím ředitele úřadu ministr a náměstci … ukládají úkoly úředníkům.;
Pokyn ministerstva obrany ukládal složkám resortu přísný zákaz ničit po výcviku munici.
pf: Vladimir Putin uložil vládě úkol něco se situací dělat už před osmi lety.;
… řidičce městský úřad uložil zákaz činnosti spočívající v zákazu řízení motorových vozidel.
....
-lvc3: omezení-1, omezení-2, opatření-1, pokání-1, povinnost-1
-map3: ADDRv-ACTn
-instig3: ACT
-example3:
impf: Nová technická pravidla ukládají odstranění či omezení některých počítačových systémů.;
Současná společnost ukládá člověku početná omezení.;
ČNB se musí striktně držet zákonných postupů a může ukládat opatření k nápravě.;
… také počítač provádí hříšníka úvodními modlitbami, ptá se na jeho hříchy a ukládá pokání.;
Soud zpravidla ukládá radnicím povinnost zajistit nájemníkům náhradní bydlení.
pf: Také ruská státní agentura Rosselkhoznadzor … si vymínila právo uložit omezení dovozu, …;
Soud může pachateli uložit omezení, aby se zdržel řízení.;
Proto jsme také městským službám neuložili žádné zvláštní opatření.;
Zřejmě to ale nebyl jen tak ledaskdo, protože sám papež uložil Bernardu Ignácovi pokání.;
Soud uložil žalované společnosti povinnost zaplatit kanceláři zmíněnou částku s příslušenstvím.

Figure 8. Simplified lexical unit of the light verb 'uložit' 'to give'.

than one set of these attributes is relevant with a single lexical unit of a light verb, the sets are distinguished by Arabic numerals. (iv) In addition, each LVC is exemplified by corpus example provided in the attribute `example`, attached to individual sets of the above given attributes. See the example of the lexical unit of the light verb *uložit* 'to give' in Figure 8.

As a result, 117 lexical units characterizing light verbs have been annotated; these lexical units are contained in 78 verb lexemes, represented by 145 verb lemmas, see Table 2. Most light verbs are non-causative: from 117 lexical units in total, 89 underlie non-causative light verbs in the annotated data and 28 lexical units correspond to causative light verbs. The semantic participant 'Causator' has been mapped predominantly onto ACT of light verbs, in less cases it has corresponded to ORIG. For the basic statistics on the mapping of 'Causator' see Table 3.

'Causator'	Lexical Units	LVCs
ACT	25	325 (365)
ORIG	3	13 (13)

Table 3. The basic statistics on the mapping of 'Causator' with causative light verbs in VALLEX; the first column provides the number of lexical units representing causative light verbs, the second one gives the number of LVCs in which light verbs are of causative character (counted as combinations of individual lexical units of light verbs and lexical units of predicative nouns). The number in parenthesis includes also LVCs with nouns that combine only with one light verb.

5.3. Types of Coreference

As a result of the annotation, almost 1,500 different LVCs, counted as combinations of individual lexical units of light verbs and lexical units of predicative nouns, have been identified. Each LVC is characterized by a certain type of coreference between verbal and nominal valency complementations (or by more than one type), see Section 3.1.4. The information on coreference has been assigned to the valency frame of a light verb (attribute `map`). For causative light verbs, also the information on 'Causator' mapping is provided (attribute `instig`).

In the annotated data, most LVCs are characterized by a single type of coreference and (if relevant) by the mapping of 'Causator'; however, almost 200 annotated LVCs allow for different types of coreference, often distinguished by the presence/absence of 'Causator'.

In the annotation, 21 different types of coreference (distinguished with respect to the presence of 'Causator') have been identified for all LVCs (including 7 ambiguous types). The most frequent type of coreference is represented by the coreference of the ACT of a non-causative light verb with the ACT of a predicative noun; this type can be exemplified by, e.g., the LVC *mít obavu* 'to be afraid; lit. to have fear' (11). Within the group of causative light verbs, the coreference of the verbal ADDR and the nominal ACT with the mapping of 'Causator' onto the verbal ACT is the most frequent one; it characterizes, e.g., the LVC *přinést poznatek* 'to bring knowledge' (35). Table 4 provides all types of coreference identified in the annotated data, illustrated by examples.

6. Conclusion

In this paper, we have summarized results of a theoretical analysis of syntactic behavior of Czech light verb constructions. We have focused both on their deep and surface syntactic structure, demonstrating the syntactic compositionality of these constructions. We have deepened an insight into a key role of coreference between valency complementations of the light verb and the predicative noun forming an LVC

Type 1 'Causator'	Type 1 Coreference	Type 2 'Causator'	Type 2 Coreference	Number	Example
	ACT_V–ACT_N			716 (902)	*udělat chybu* 'to make a mistake'
	ACT_V–ACT_N, $ADDR_V$–$ADDR_N$			182 (200)	*činit výtku* 'make a reproach'
	ACT_V–PAT_N, LOC_V–ACT_N	ACT_V	LOC_V–ACT_N	141 (151)	*vyvolat strach* 'to raise fear'
	ACT_V–ACT_N, $ADDR_V$–PAT_N			99 (142)	*klást odpor* 'put up resistance'
ACT_V	$ADDR_V$–ACT_N			68 (82)	*poskytnout útěchu* 'to give comfort'
	ACT_V–$ADDR_N$, $ORIG_V$–ACT_N			60 (67)	*přijmout poděkování* 'to accept thanks'
ACT_V	LOC_V–ACT_N			60 (65)	*vzbouzet myšlenky* 'to provoke thoughts'
	ACT_V–PAT_N, $ORIG_V$–ACT_N			35 (44)	*dostat péči* 'to get care'
	ACT_V–PAT_N, $ADDR_V$–ACT_N	ACT_V	$ADDR_V$–ACT_N	35 (44)	*činit potěšení* 'to make pleasure'
	ACT_V–PAT_N, LOC_V–ACT_N			30 (37)	*nacházet oporu* 'to find support'
	ACT_V–ACT_N, LOC_V–PAT_N			13 (17)	*nalézt zalíbení* 'to develop a taste'
$ORIG_V$	ACT_V–ACT_N			13 (13)	*získat výhodu* 'to gain an advantage'
ACT_V	BEN_V–ACT_N			8 (10)	*otevřít přístup* 'to open an access'
	ACT_V–ACT_N		ACT_V–PAT_N, LOC_V–ACT_N	8 (8)	*ztratit důvěru* 'to lose confidence'
	ACT_V–$ORIG_N$, LOC_V–ACT_N			7 (7)	*budit dojem* 'to give an impression'
	ACT_V–$ADDR_N$, LOC_V–ACT_N			7 (7)	*udělat zkoušku* 'to pass an exam'
	ACT_V–$ORIG_N$, LOC_V–ACT_N	ACT_V	LOC_V–ACT_N	6 (6)	*probouzet pocit* 'to inspire a feeling'
	ACT_V–$ADDR_N$, LOC_V–ACT_N	ACT_V	LOC_V–ACT_N	4 (4)	*vzbudit podezření* 'to raise suspicion'
ACT_V	$ADDR_V$–PAT_N, LOC_V–ACT_N			2 (2)	*přinést přízeň* 'to bring favor'
	ACT_V–$ADDR_N$, LOC_V–ACT_N		ACT_V–PAT_N, LOC_V–ACT_N	1 (1)	*budit soustrast* 'to arouse sympathy'
	ACT_V–ACT_N	ACT_V	BEN_V–ACT_N	1 (1)	*vytvářet zisk* 'to make a profit'

Table 4. The basic statistics on the coreference identified with the annotated LVCs (individual combinations of lexical units of light verbs and lexical units of predicative nouns), taking into account the type of coreference and the 'Causator' mapping (if relevant). If more than one type is characteristic of a single LVC, these types are distinguished as type 1 and type 2. The number in column 5 gives the number of LVCs in VALLEX; in parenthesis, the number including also those LVCs with nouns combining with a single light verb follows.

in the process of the syntactic formation of the given LVC, emphasizing a role of mapping of semantic participants characterizing the given LVC onto valency complementations.

The proposed theoretical analysis has been verified (and refined) within a linguistic annotation of a large amount of light verb constructions. The resulting description has been used in the VALLEX lexicon for their theoretically adequate and economic representation. The data have been published – after both manual and automatic data consistency checking –- in the new version of the VALLEX lexicon, release 3.5.[14]

The VALLEX 3.5 lexicon comprises annotation of almost 3,000 collocations of predicative nouns with light verbs (counted as combinations of a lemma of a light verb and a lemma of a predicative noun), which correspond to almost 1,500 LVCs (counted as individual combinations of a lexical unit of a light verb and a lexical unit of a predicative noun). The LVC annotation has affected almost 350 newly created lexical units of predicative nouns and 120 lexical units of light verbs. The syntactic formation of LVCs has been described in the grammar component of VALLEX in a form of syntactic rules operating on the information from the data component (namely valency frames of respective light verbs and predicative nouns and three special attributes describing the LVC formation `lvc`, `map`, and `instig`).

Acknowledgements

The work on this project has been supported by the grant of the *Czech Science Foundation* (project GA15-09979S); the data annotation has been partially supported by the LINDAT/CLARIN project of the *Ministry of Education, Youth and Sports of the Czech Republic* (project number LM2015071).

This work has been using language resources distributed by the LINDAT/CLARIN project of the *Ministry of Education, Youth and Sports of the Czech Republic* (project number LM2015071).

Bibliography

Alonso Ramos, Margarita. Towards the Synthesis of Support Verb Constructions: Distribution of Syntactic Actants between the Verb and the Noun. In Wanner, L. and I. A. Mel'čuk, editors, *Selected Lexical and Grammatical Issues in the Meaning-Text Theory*, pages 97–137. John Benjamins Publishing Company, Amsterdam, Philadelphia, 2007.

Bejček, Eduard, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. *Prague Dependency Treebank 3.0*. Univerzita Karlova v Praze, MFF, ÚFAL, Prague, Czech Republic, 2013. `http://ufal.mff.cuni.cz/pdt3.0`.

[14] `http://ufal.mff.cuni.cz/vallex/3.5/`

Butt, Miriam. The Light Verb Jungle: Still Hacking Away. In Amberber, Mengistu, Brett Baker, and Mark Harvey, editors, *Complex Predicates in Cross-Linguistic Perspective*, pages 48–78. Cambridge University Press, Cambridge, 2010.

Butt, Miriam and Wilhelm Geuder. *On the (semi)lexical status of light verbs*, volume 59 of *Studies in Generative Grammar*, pages 323–370. Mouton de Gruyter, Berlin – New York, 2001.

Grimshaw, Jane and Armin Mester. Light verbs and θ-marking. *Linguistic inquiry*, 19(2):205–232, 1988.

Hinrichs, Erhard, Andreas Kathol, and Tsuneko Nakazawa. *Complex Predicates in Nonderivational Syntax. Syntax and Semantics 30*. Academic Press, San Diego, 1998.

Jespersen, Otto. *A Modern English Grammar on Historical Principles, Part VI, Morphology*. Allen and Unwin, 1965.

Kettnerová, Václava. Syntaktická struktura komplexních predikátů. *Slovo a slovesnost*, 18(1): 3–24, 2017.

Kettnerová, Václava and Eduard Bejček. Distribution of Valency Complements in Czech Complex Predicates: Between Verb and Noun. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 515–521, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Kettnerová, Václava and Markéta Lopatková. The Representation of Czech Light Verb Constructions in a Valency Lexicon. In Hajičová, Eva, Kim Gerdes, and Leo Wanner, editors, *Proceedings of the Second International Conference on Dependency Linguistics, Depling 2013*, pages 147–156, Praha, Czechia, 2013. Univerzita Karlova v Praze, Matfyzpress. ISBN 978-80-7378-240-5.

Kettnerová, Václava and Markéta Lopatková. At the Lexicon-Grammar Interface: The Case of Complex Predicates in the Functional Generative Description. In Hajičová, Eva and Joakim Nivre, editors, *Proceedings of Depling 2015*, pages 191–200, Uppsala, Sweden, 2015. Uppsala University.

Kettnerová, Václava and Markéta Lopatková. Complex Predicates with Light Verbs in VALLEX: From Formal Model to Lexicographic Description. In Hlaváčová, Jaroslava, editor, *Proceedings of the 17th Conference on Information Technologies - Applications and Theory (ITAT 2017)*, volume 1885 of *CEUR Workshop Proceedings*, pages 15–22, Praha, Czechia, 2017a. ÚFAL MFF UK, CreateSpace Independent Publishing Platform.

Kettnerová, Václava and Markéta Lopatková. Ke koreferenci u komplexních predikátů s kategoriálním slovesem. *Korpus – gramatika – axiologie*, 16:3–26, 2017b.

Kettnerová, Václava, Petra Barančíková, and Markéta Lopatková. Lexicographic Description of Complex Predicates in Czech: Between Lexicon and Grammar. In Margalitadze, Tinatin and George Meladze, editors, *Proceedings of the XVII EURALEX International Congress: Lexicography and Linguistic Diversity*, Tbilisi, Georgia, 2016. Ivane Javakhishvili Tbilisi State University, Tbilisi University Press.

Kettnerová, Václava, Veronika Kolářová, and Anna Vernerová. Deverbal Nouns in Czech Light Verb Constructions. In Mitkov, Ruslan, editor, *Lecture Notes in Artificial Intelligence, Computational and Corpus-Based Phraseology. Second International Conference, Europhras 2017. London, UK, November 13???14, 2017.*, volume 10596 of *Lecture Notes in Computer Science*, pages 205–219, Cham, Switzerland, 2017. European Association for Phraseology EUROPHRAS, University of Wolverhampton, Association for Computational Linguistics, Bulgaria, Springer.

Kilgarriff, Adam, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. The Sketch Engine: ten years on. *Lexicography ASIALEX*, 1(1): 7–36, 2014.

Lopatková, Markéta and Jarmila Panevová. Recent Developments in the Theory of Valency in the Light of the Prague Dependency Treebank. In Šimková, Mária, editor, *Insight into Slovak and Czech Corpus Linguistics*, pages 83–92. Veda, Bratislava, 2006.

Lopatková, Markéta, Václava Kettnerová, Eduard Bejček, Anna Vernerová, and Zdeněk Žabokrtský. *Valenční slovník českých sloves VALLEX*. Karolinum, Praha, 2016.

Panevová, Jarmila. On Verbal Frames in Functional Generative Description I–II. *The Prague Bulletin of Mathematical Linguistics*, 22–23:3–40, s. 17–52, 1974–75.

Panevová, Jarmila. *Formy a funkce ve stavbě české věty*. Academia, Praha, 1980.

Panevová, Jarmila. Valency Frames and the Meaning of the Sentence. In Luelsdorff, Philip A., editor, *The Prague School of Structural and Functional Linguistics*, pages 223–243. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1994.

Radimský, Jan. *Verbo-nominální predikát s kategoriálním slovesem*. Editio Universitatis Bohemiae Meridionalis, České Budějovice, 2010.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Reidel, Dordrecht, 1986.

Urešová, Zdeňka. *Valence sloves v Pražském závislostním korpusu*. Ústav formální a aplikované lingvistiky, Praha, 2011.

Address for correspondence:
Václava Kettnerová
kettnerova@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 57–86

PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing

Lauriane Aufrant,[a b] Guillaume Wisniewski[b]

[a] DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
[b] LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France

Abstract

We present PanParser, a Python framework dedicated to transition-based structured prediction, and notably suitable for dependency parsing. On top of providing an easy way to train state-of-the-art parsers, as empirically validated on UD 2.0, PanParser is especially useful for research purposes: its modular architecture enables to implement most state-of-the-art transition-based methods under the same unified framework (out of which several are already built-in), which facilitates fair benchmarking and allows for an exhaustive exploration of slight variants of those methods. PanParser additionally includes a number of fine-grained evaluation utilities, which have already been successfully leveraged in several past studies, to perform extensive error analysis of monolingual as well as cross-lingual parsing.

1. Introduction

PanParser is not yet another implementation of a transition-based dependency parser. Transition-based dependency parsing has been an active field in the last few years and several open source parsers have been released, each one implementing a new alternate paradigm, like MaltParser (Nivre et al., 2006a), which is the reference implementation for transition-based parsers, and UDPipe (Straka and Straková, 2017), a popular pipeline system for neural parsing. In all transition-based parsers, the same elements are systematically found (a transition system, a classifier, an update procedure, etc.), corresponding to distinct lines of research. However, their implementations often adopt an ad-hoc architecture or a specific variation of each of these

 Corresponding author: lauriane.aufrant@limsi.fr
Cite as: Lauriane Aufrant, Guillaume Wisniewski. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 57–86.
doi: 10.2478/pralin-2018-0007.

components, which impedes fair benchmarking and makes it difficult to evaluate the impact of a given component.

The PanParser framework aims at alleviating this effect by providing a modular architecture, in which most state-of-the-art transition-based parsing systems can be implemented, or extended, in a light and straightforward way. In addition to providing an easy way to train accurate models for parsing any language, it is then particularly valuable for research purpose and exhaustive experiments on parser design. It is possible, for instance, to train a greedy neural ArcEager parser with a static oracle, or a delexicalized beam ArcHybrid parser with a dynamic oracle and an averaged perceptron.[1] PanParser also implements extensive utilities for error analysis. PanParser has been used, for instance, in cross-lingual transfer experiments (Aufrant et al., 2016b; Lacroix et al., 2016) or to design new learning strategies for dependency parsers (Aufrant et al., 2017).

PanParser differs from the other parsing frameworks by the ability to combine more freely alternate versions of each part of the parser, but also by the diversity of the built-in algorithms: for instance, UDPipe does not include the ArcEager and ArcHybrid transition systems and it lacks the support of global training strategies, spaCy[2] only offers ArcEager parsers trained with the max-violation strategy, while SyntaxNet (Andor et al., 2016) and the StanfordParser (Chen and Manning, 2014) focus on the ArcStandard system; as for MaltParser, it supports a large number of transition systems but neither global training nor dynamic oracles. PanParser also includes functionalities that are not found elsewhere, like newly-derived dynamic oracles or the ability to train projective parsers directly on non-projective data. On the other hand, our framework is not designed for pipelining as others are; the current built-ins also lack several non-projective transition systems included in UDPipe and MaltParser, although the architecture is already designed to support them – and recent works like Fernández-González and Gómez-Rodríguez (2018a)'s will help their implementation in future work. As for graph-based parsing strategies, like the seminal MSTParser (McDonald et al., 2005) or state-of-the-art neural ones (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017), they remain out of the scope of this framework.

The whole software is written in Python in a modular way, which makes it easy for the user to extend the built-in components with custom variants. For instance, adding the ArcHybrid transition system (with full compatibility with all other components) was done in 150 lines of code. The core framework of PanParser can also be reused to implement other structured prediction tasks, as done for the built-in PoS tagger.

The rest of the paper is organized as follows. Section 2 presents the state of the art in transition-based dependency parsing, along the six lines of research which have

[1] See Section 2 and Supplementary A for a brief description of those algorithms.

[2] `https://spacy.io`

guided the design of PanParser. The main features of PanParser are described in Section 3 and Section 4 presents the scores achieved on the 73 treebanks of UD 2.0, for various configurations of PanParser. Further algorithmic and technical details are provided in the appendices and supplementary material: the procedure used to implement dynamic oracles for various transition systems in PanParser (Appendix A), the global dynamic oracle framework on which PanParser is based (Appendix B), as well as the built-in transition systems with their oracles (Supplementary A), practical examples of PanParser usage (Supplementary B) and a brief overview of the code architecture chosen to ensure modularity (Supplementary C).

PanParser is published under a BSD license and can be freely downloaded at `https://perso.limsi.fr/aufrant`.

2. Transition-based dependency parsing

Dependency parsing (Kübler et al., 2009) consists in analyzing the syntactic structure of a sentence by mapping it to a tree. Formally, a unique head token is assigned to each token of the sentence (apart from the root), avoiding cycles, to denote syntactic dependencies between two words; in labeled parsing, a relation label is additionally assigned to each token. In this section, we introduce one class of algorithms building dependency trees, the transition-based approach (Nivre, 2008), which is the one adopted by PanParser.

In a transition-based parser, a parse is computed by performing a sequence of *transitions*, building the parse tree in an incremental fashion. The *parser configuration* thus represents a partially built dependency tree, and applying transition t to configuration c results in the parser moving to a *successor* configuration of c (denoted $c \circ t$), together with side effects on its internal state (typically based on stacks and lists): moving a token, creating an edge, etc.

According to Nivre (2008), '*a deterministic classifier-based parser consists of three essential components: a parsing algorithm, which defines the derivation of a syntactic analysis as a sequence of elementary parsing actions; a feature model, which defines a feature vector representation of the parser state at any given time; and a classifier, which maps parser states, as represented by the feature model, to parsing actions*'. Over the years, much work has been dedicated to improving parsers along those three lines: designing new parsing algorithms (i.e. transition systems, in the case of transition-based parsing) with various properties (Nivre, 2004, 2009; Gómez-Rodríguez and Nivre, 2010; Kuhlmann et al., 2011; Qi and Manning, 2017), more informative feature representations (Zhang and Clark, 2008; Zhang and Nivre, 2011; Bohnet et al., 2013; Alberti et al., 2015), and adapting the implementations to use more accurate classifiers (Chen and Manning, 2014; Dyer et al., 2015; Zhou et al., 2015; Andor et al., 2016).

However, in recent years, a series of contributions has been made in the way those three components interact, in particular at training time (Collins and Roark, 2004; Zhang and Clark, 2008; Goldberg and Nivre, 2012; Huang et al., 2012, Zhang and Nivre,

2012, Aufrant et al., 2017). The resulting lines of research have produced a number of algorithmic variants, with such diversity that we now find it beneficial to model these aspects as separate components. Therefore, the new reading grid we propose, and which has been adopted as the architecture of PanParser, is the following: a transition-based parser consists of 6 components (a transition system, a classifier, a feature model, a search component, an oracle and a training strategy), whose interactions can be controlled by a generic structured prediction framework, with no specific tie to the chosen algorithm or task.

In the following, we describe the state-of-the-art methods corresponding to each component, out of which most are or can be implemented under the PanParser framework. The actual built-ins will be listed in the next section, while the other algorithms can still be implemented within that framework, thanks to its modular architecture.

2.1. Transition system

The transition system defines the semantics of the actions predicted by the classifier, in order to relate them to actual dependency trees. Formally, it consists in four elements: the structure of the parser state (e.g. a stack and a buffer), a set of actions (e.g. Shift, Left, Right and Reduce), the semantics of each one (e.g. the Left action attaches the current token to another token on the right, and then discards it from the stack) and their preconditions (e.g. the Left action is invalid if the current token has already been attached).

Several such systems have been proposed in the literature, the most widely used ones being stack-based. Notably, ArcStandard (Nivre, 2003), ArcEager (Nivre, 2004) and ArcHybrid (Kuhlmann et al., 2011) rely on a stack (the currently processed tokens) and a buffer (the not yet processed tokens). They all consider candidate edges among the top few tokens on the stack and the first few tokens in the buffer, but they differ in the order in which edges are created, and regarding spurious ambiguity (i.e. whether the same tree can be produced by different derivations). All three systems guarantee parsing in linear time, but they are also subject to restrictions in expressivity (as they cannot produce non-projective trees, i.e. trees with crossing edges).

In order to improve expressivity or to facilitate the prediction of some edges, a number of other systems have been designed, using various kinds of internal state: a stack and a list (Nivre, 2009; Fernández-González and Gómez-Rodríguez, 2012), two lists and a buffer (Covington, 2001; Choi and Palmer, 2011), a single list (Goldberg and Elhadad, 2010), two (or more) stacks and a buffer (Gómez-Rodríguez and Nivre, 2010; Gómez-Rodríguez and Nivre, 2013), extra registers on top of the stack and the buffer (Pitler and McDonald, 2015), etc.

Many of those systems have seen additional variants, adding for instance non-monotonicity (Honnibal et al., 2013; Honnibal and Johnson, 2015; Fernández-González and Gómez-Rodríguez, 2017) to improve the robustness to erroneous predictions, or non-local transitions (Qi and Manning, 2017, Fernández-González and Gómez-

Rodríguez, 2018b), effectively shortening the derivations by collapsing series of Shift or Reduce actions into edge-creating transitions. The ability to parse given specific constraints on the output has also been investigated as a property of the transition system (Nivre and Fernández-González, 2014; Nivre et al., 2014).

Finally, several variants of each of those systems exist natively, depending on how the root tokens are handled. Indeed, the roots are supposed to remain unattached at the end of the derivation, but in order to alleviate boundary effects they are in general attached to a dummy Root token placed at the beginning or the end of the sentence. Yet, the position of this node can actually impact the semantics of the transition system, and it has been shown to impact the parsing accuracy (Ballesteros and Nivre, 2013).

2.2. Classifier

At each step of the parsing process, every possible transition is scored by a classifier, based on a feature representation of the current configuration, and the transitions to apply are chosen accordingly. This scoring step can be handled by any multi-class classifier, and several options have been envisioned in the literature.

Linear models have notably proved successful: Collins and Roark (2004) use an averaged perceptron – meaning that the final parameters retained after training are obtained by averaging over all values taken by the model throughout training – which has long remained a *de facto* standard (Huang et al., 2012). Another widespread strategy is to use support vector machines (Nivre et al., 2006b). A number of other techniques have been considered, like memory-based learning (Nivre et al., 2004), robust risk minimization (Choi and Nicolov, 2009) and confidence-weighted classifiers (Haulrich, 2010), but the largest body of recent work on the topic concerns the integration of neural networks, with various architectures: feedforward neural networks (Chen and Manning, 2014; Zhou et al., 2015), later augmented by a structured perceptron (Weiss et al., 2015) or a CRF loss (Andor et al., 2016), as well as recurrent networks (Stenetorp, 2013; Chen et al., 2015; Dyer et al., 2015). However, recurrent neural networks (and notably, LSTMs) cannot be implemented in the current version of the PanParser framework, which supports only stateless classifiers.

2.3. Feature model

The feature model specifies how the parser configuration is represented when it is fed to the classifier. Such features are typically extracted from the top of the stack and buffer: wordform of the first token in buffer, PoS tag of the (already attached) children of the stack head, etc. In addition to wordforms, morphological features and coarse or fine-grained PoS tags, richer features like sequential (token distance), syntactic (valency, labels, representation of subtrees) or semantic information (semantic classes, pre-trained word embeddings) can also be collected for any such token (Zhang and Nivre, 2011; Agirre et al., 2011; Chen and Manning, 2014; Dyer et al., 2015).

Standard templates	
1 word	w, p and wp for S_0, N_0, N_1, N_2
2 words	wp·wp, wp·w, w·wp, wp·p, p·wp, w·w and p·p for S_0·N_0; N_0p·N_1p
3 words	p·p·p for N_0·N_1·N_2, S_0·N_0·N_1, S_{0h}·S_0·N_0, S_0·S_{0l}·N_0, S_0·S_{0r}·N_0, S_0·N_0·N_{0l}
New templates with rich non-local features	
Distance	S_0w·d, S_0p·d, N_0w·d, N_0p·d; S_0w·N_0w·d, S_0p·N_0p·d
Valency	S_0wv_l, S_0pv_l, S_0wv_r, S_0pv_r, N_0wv_l, N_0pv_l
Unigrams	w and p for S_{0h}, S_{0l}, S_{0r}, N_{0l}; l for S_0, S_{0l}, S_{0r}, N_{0l}
Third-order	w and p for S_{0h2}, S_{0l2}, S_{0r2}, N_{0l2}; l for S_{0h}, S_{0l2}, S_{0r2}, N_{0l2}; p·p·p for S_0·S_{0h}·S_{0h2}, S_0·S_{0l}·S_{0l2}, S_0·S_{0r}·S_{0r2}, N_0·N_{0l}·N_{0l2}
Label set	S_0ws_l, S_0ps_l, S_0ws_r, S_0ps_r, N_0ws_l, N_0ps_l

Table 1: Feature templates proposed by Zhang and Nivre (2011). S_0 is the top stack element, N_0-N_2 the 3 first buffer elements. w, p and l stand for word, PoS tag and label. d is the token distance from S_0 to N_0. v_l (v_r) is the number of left (right) children, whose labels are s_l (s_r). $_h$ and $_{h2}$ denote the head and its own head, $_l$, $_{l2}$ ($_r$, $_{r2}$) the first and second leftmost (rightmost) children.

The resulting feature vector representation can be directly fed to non-linear classifiers like kernel SVMs and neural networks, while for linear models *feature templates* are typically used to combine the extracted atomic features into tuple features. Table 1 describes the templates handcrafted by Zhang and Nivre (2011), which are known for achieving state-of-the-art performance on several languages, while others advocate for automatic selection (Nilsson and Nugues, 2010; Ballesteros and Nivre, 2012; Ballesteros and Bohnet, 2014).

It can however be noted that depending on the exact parser settings, not all this information can be successfully extracted. For instance the delexicalized parsers used in cross-lingual transfer (Zeman and Resnik, 2008) make no use of wordforms, unlabeled parsers exclude label information, and depending on the transition system, some syntactic features remain unknown at prediction time (in bottom-up systems like ArcStandard and ArcHybrid the head of stack elements is never known, while in ArcEager it can be).

2.4. Search

Both during training and prediction, parsing is done by exploring the search space composed by all transitions scored by the classifier, in search for the best possible tree. Transition sequences are scored by summing the scores of all their transitions; parsing thus amounts to finding the derivation having the highest score. However, this inference step is hindered by the exponential size of the search space.

While exact inference can be done through dynamic programming (Huang and Sagae, 2010; Kuhlmann et al., 2011; Zhao et al., 2013), such methods imply severe restrictions on the set of authorized features, which jeopardize the accuracy of the parser. Instead, inexact search is generally employed, either as greedy or beam search. Greedy decoding (i.e. always following the single-best transition) is faster, but beam search (Zhang and Clark, 2008) yields more accurate parsers as it explores a larger part of the space: it maintains a set of k parse candidates (the beam) and at each step all possible actions are considered for each hypothesis in the beam, after which only the k-best resulting configurations are kept. This method faces some computational issues due to memory management, but they can be easily avoided by using tree-structured stacks and distributed representations of trees (Goldberg et al., 2013).

While their implementations often differ, it can be noted that greedy search is mathematically equivalent to beam search with $k = 1$. As for exhaustive search, it can be modeled with a beam of infinite size.

2.5. Oracle

Training of transition-based parsers is a two-step procedure: first some decoding is performed (using the chosen search strategy), then whenever an error a detected, the model is updated based on one (or more) predicted configuration(s) and one (or more) gold configuration(s).

The oracle is the component that governs the distinction between gold and erroneous configurations. More precisely, its role at training time is twofold: flagging errors during decoding and identifying gold configurations that can serve as positive (reference) configurations for updates, while the actual choice of update configurations depends on the training strategy.

The traditional and most straightforward kind of oracle is the static one. In a static oracle, references are precomputed heuristically, based on the semantics of the transition system: for each reference tree used as example, a unique derivation leading to that tree is chosen and its transitions become references, while all others are considered erroneous. While simple to implement, static oracles have two drawbacks: they ignore spurious ambiguity (when several derivations lead to the same reference tree, all but one are considered erroneous) and they are only defined along the reference derivation (given an arbitrary configuration in the search space, the optimal action to take next is not specified).

Instead, Goldberg and Nivre (2012) introduce dynamic oracles, defined as oracles that are both non-deterministic and complete: with dynamic oracles, the reference actions are tailored to the current configuration. In that framework, erroneous actions are defined as actions that introduce errors, i.e. that reduce the maximum score achievable on the current sentence (for instance by misattaching a token, or by removing from the stack a word which has not received all its children yet). The computation of the oracle is based on the *action cost*, which is a property of the transition system

and numbers the errors introduced by each action, given an arbitrary configuration. The gold actions are defined as zero-cost actions; this way the oracle is guaranteed to find the best action(s) to perform, and it natively accounts for spurious ambiguity (an action leading to the reference tree is by design zero-cost) and is always defined (by definition of the maximum there is always at least one zero-cost action).

The main difficulty when applying dynamic oracles is to derive the action cost for the transition system, which requires a thorough investigation of its properties. So far, such oracles have been derived for ArcEager (Goldberg and Nivre, 2012), ArcHybrid and EasyFirst (Goldberg and Nivre, 2013), ArcStandard (Goldberg et al., 2014), as well as several non-monotonic (Honnibal et al., 2013; Honnibal and Johnson, 2015) and non-projective systems (Gómez-Rodríguez et al., 2014; Gómez-Rodríguez and Fernández-González, 2015; Fernández-González and Gómez-Rodríguez, 2017). A few other works have proposed oracles drawing from this line of research but which are only partly dynamic, incomplete or approximated (Björkelund and Nivre, 2015; de Lhoneux et al., 2017; Fernández-González and Gómez-Rodríguez, 2018a). Aufrant et al. (2018) additionally propose a systematic way to approximate the oracle when it has not yet been derived to handle all particular cases – for instance for reference trees that are out of the expressive scope of the transition system, typically non-projective training examples for projective parsers – by using an unsound cost and considering the minimum-cost actions as gold. This is the approach adopted within our framework; it is further described in Appendix A, which explains in which measure PanParser departs from the standard dynamic oracle framework.

While dynamic oracles are primarily defined to identify reference *actions*, Aufrant et al. (2017) extend them to reference *transition sequences*, to accommodate their use with beam parsers. The advantage of these *global dynamic oracles* is that they do not require any explicit computation of the reference set (which can be exponentially huge), as they can be implemented by simply tracking the action costs inside the beam.

2.6. Training strategy

The literature traditionally distinguishes local and global training strategies. In local training, each decision is optimized independently: all along the derivation, the action predicted by the classifier is checked against the gold action(s) (the reference(s) provided by the oracle), and an update occurs whenever they differ.

When beam search is employed, local training is suboptimal (Zhang and Nivre, 2012) and a global criterion is preferred, meaning that the parameters are updated once for each training sentence, depending on the optimality of transition *sequences*. Algorithm 1 (left part) summarizes the training for each sentence x (with gold parse y): $\textsc{Initial}(x)$ denotes the initial configuration for x and the $\textsc{Oracle}$ procedure performs decoding to find configurations that play the role of the 'positive' and 'negative' examples (resp. c^+ and c^-) required by the $\textsc{Update}$ operation (typically a perceptron update rule (Collins and Roark, 2004) or a gradient computation with the globally

normalized loss of Andor et al. (2016)). Several strategies, corresponding to various implementations of the ORACLE function, have been used to find these examples.

Algorithm 1: Global training on one sentence, with and without restart.

θ: model parameters, initialized to θ_0 before training
FINAL$(\cdot)$: true iff the whole sentence is processed

Function *TRAINONESENTENCE(x,y)*

```
c ← INITIAL(x)

c⁺, c⁻ ← ORACLE(c, y, θ)
θ ← UPDATE(θ, c⁺, c⁻)
```

Function *TRAINONESENTENCERESTART(x,y)*

```
c ← INITIAL(x)
while ¬FINAL(c) do
    c⁺, c⁻ ← ORACLE(c, y, θ)
    θ ← UPDATE(θ, c⁺, c⁻)
    c ← c⁺
```

In the EARLYUPDATE strategy (Collins and Roark, 2004; Zhang and Clark, 2008), the sentence is parsed using conventional beam decoding, checking at each step whether the reference tree is still reachable, and an update happens as soon as the reference derivation (or all references, depending on what the oracle generates) falls off the beam: the top scoring configuration at this step is penalized and the reference that has just fallen off the beam is reinforced. Another strategy, MAXVIOLATION (Huang et al., 2012), is to continue decoding even though the reference has fallen off the beam, in order to find the configuration having the largest gap between the scores of the (partial) hypothesis and the (partial) gold derivation. Compared to EARLYUPDATE, MAXVIOLATION speeds up convergence by covering longer transition sequences and can yield slightly better parsers.

In the standard version of these strategies, after a global update the rest of the sequence is ignored, moving on to the next example. However, Aufrant et al. (2017) extend those strategies with a restart option (right part of Algorithm 1) which reinitializes the beam after each update and enables further updates on the same example, so that the whole sentence is exploited during training, with benefits in terms of convergence, accuracy and sampling distributions.

Notably, under the restart framework – and similarly to the equivalence of search strategies – local training can be interpreted as a special case of global training, when applying early update and restart on a beam of size 1.

One key aspect when choosing a training strategy is how the training configurations are generated: each update (either local or global) raises questions on which configuration to restart from, the positive or the negative one. Goldberg and Nivre (2012) show that error exploration, i.e. pursuing on the erroneous path, improves the accuracy by making the classifier able to produce the next best tree, even when the op-

timal one has become unreachable. Classifiers that stick to the gold space at training time suffer indeed from error propagation, as the suboptimal configurations they are confronted to at prediction time are an unknown territory in which they were never trained to take good decisions. As a trade-off between strict supervision and robustness, various exploration policies can be envisioned; for instance Goldberg and Nivre (2013) keep the first iteration in the gold space, and then apply error exploration with a probability of 90%, while Ballesteros et al. (2016) sample the next configuration from the probability distribution output by the classifier. However, error exploration requires oracle completeness, and can thus only be entertained when using a dynamic oracle.

3. The PanParser implementation

As described in the previous section, a transition-based parser can be viewed as the association of several components: a *transition system* (associating parse trees with transition sequences), a *classifier* (scoring transitions based on a feature representation), a *feature model* (extracting feature vectors from parse configurations), a *search strategy* (producing transition sequences, given a classifier model), an *oracle* (mapping gold annotations to gold transitions) and a *training strategy* (effectively choosing the training configurations to update the model).

In PanParser, to ensure modularity and compatibility, all of these components are implemented separately – and interfaced by a generic structured prediction framework, which handles the main training and prediction logic. The first three correspond to distinct modules, for which we provide several implementations, and which can be easily extended by alternate models or implementations. The definition of the *transition system* includes all properties that are system-specific, so that the other modules can be fully semantics-agnostic: its API can be queried to return (given a configuration) the list of valid actions, the action costs, a successor configuration after a given action, the partial tree already built (as an on-demand computation based on transition history), and whether the state is final. The *classifier* is also seen as a black box by the rest of the software, with a score/predict/update API (plus some initialization functions) that makes it possible to integrate any stateless classifier; support for stateful classifiers (like LSTMs) is not yet included but is planned for future work as an API extension. As for the *feature model*, it relies on high-level properties of the parse configurations (i^{th} word in buffer, head of the top of the stack, etc.) to generate feature representations, either as atomic or templated features (to accommodate both linear and non-linear classifiers).

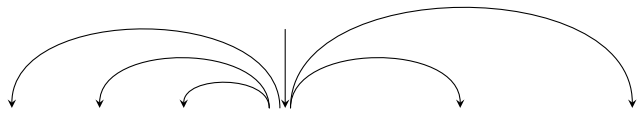
The three other components (search, oracle and training strategy) are based on an extensive set of built-in parameters which can be set and combined at will. Following Aufrant et al. (2016a), the whole search/learn procedure is based on beam search and global dynamic oracles, from which all other strategies (greedy search, static oracle, local training) are derived as special cases.

On top of algorithmic analyses, keeping those components independent has also required specific implementation choices and abstraction layers; more details on that matter are provided in Supplementary C.

After a brief description of how dependency trees are represented internally (§3.1), the built-in components of PanParser are described in §3.2. Further technical details are then provided on other functionalities of PanParser: enriched inputs (§3.3), parser ensembling (§3.4), a built-in PoS tagger (§3.5) and the error analysis tools (§3.6). See Supplementary B for an illustration of how these functionalities can be used and combined.

3.1. Representation of a dependency tree

Since there is a one-to-one correspondence between child tokens and dependency edges, a dependency tree can be easily modeled as a list of head tokens with padding elements (a dummy Root token or symbols denoting the start and end of sentences). In PanParser, it is represented as a list of integers, the integer at position i corresponding to the index of the head governing the i^{th} word in the sentence. Relation labels can similarly be represented as a list of strings, for labeled parsing. Figure 1 illustrates the resulting representation of a tree, with three variants depending on the position of the Root token (None, First and Last, as empirically compared by Ballesteros and Nivre (2013)); in PanParser the last position is used by default, but the first position can be set to be used internally – parsing without Root token is currently not supported.



	You^0	're^1	not^2	thinking^3	fourth-dimensionally^4	!^{5}	
	3	3	3	None	3	3	
ROOT^0	You^1	're^2	not^3	thinking^4	fourth-dimensionally^5	!^{6}	
None	4	4	4	0	4	4	
<start>^{0}	You^1	're^2	not^3	thinking^4	fourth-dimensionally^5	!^{6}	ROOT^7
None	4	4	4	7	4	4	0
None	nsubj	aux	advmod	root	advmod	punct	None

Figure 1: Dependency tree representations with various Root positions.

3.2. Built-in components

Transition system PanParser implements four transition systems (ArcEager, ArcHybrid, ArcStandard and NonMonotonicArcEager) as well as variants for partial output and short-spanned dependencies. For all, both versions with the Root token in leading and trailing positions are implemented. Some experimental options are also included, like adding head direction constraints to ArcEager.

High-level functions (extracting all possible atomic features, enumerating stack tokens, etc.) are available to facilitate the implementation of other transition systems based on a stack and a buffer; for other state structures they remain to be implemented.

A formal description of all built-in systems and their action costs is provided in Supplementary A. For each one, the soundness of action costs has been experimentally validated by exhaustive search from all configurations on all possible trees for sentences under 10 tokens.

Classifier The default classifier used by PanParser is the multi-class averaged (structured) perceptron, following Collins and Roark (2004). Support for neural networks is also included, and a vanilla feedforward neural network (based on Keras and Theano) is implemented as a proof of concept.

Additionally, generic classifiers are provided for joint prediction and voting, which are in fact wrappers around other classifiers. Joint classifiers are natively used for instance to enable labeled parsing: based on the same feature representations, actions and relation labels are predicted in parallel by two classifiers (which may or may not share parameters, depending on the implemented classifiers). As for the voting wrapper, it enables parser ensembling with (weighted) votes at the action level.

Feature model PanParser is shipped with the feature templates of Zhang and Nivre (2011), together with options that extend them with morphological features and pretrained embeddings; it is also possible to use the atomic version of the features (for non-linear classifiers) or to write custom templates, based on the provided atomic features (including both coarse and fine-grained PoS tags). In any case, two global parameters control whether lexicalized features and label information are included (to build delexicalized and unlabeled parsers).

Search As explained above, PanParser allows both greedy and beam search, the former being a special case of the latter; it consequently implements the data structure introduced for beam search by Goldberg et al. (2013), based on immutable objects and distributed representations. This implementation is, however, suboptimal for vanilla greedy parsers, which can be optimized using mutable objects, so that we also provide an alternate implementation dedicated to greedy parser states.

The mutable version represents the state of the parser as a buffer pointer, a stack and a parse tree, that get updated whenever a transition is applied. This structure

makes much information available in constant time (which notably speeds up feature extraction), but it cannot be used for beam search without costly object copies (Goldberg et al., 2013) and has consequently limited functionalities. The mathematical equivalence of both implementations for greedy parsing has been experimentally validated.

In Goldberg et al. (2013)'s version, which is the default, a parser state is just an immutable set of a few indexes and pointers to other parser states (previous state and tail of the stack).[3] Thus, derivations are represented as linked lists, and the complete information about a parser state (content of the stack, transition history, current parse tree) is distributed across all previous states, without duplicates. While accessing a deep stack element is necessarily slower than in the local implementation, factoring information in this way makes beam search and global training cheaper, both in time and memory usage.

Oracle Both static and dynamic oracles (including global dynamic ones) are supported by PanParser; static oracles are actually computed using dynamic ones – the reference derivation is built by pre-parsing the sentence while restricting the search space to zero-cost actions and ignoring their score.

Training strategy PanParser supports both local and global training, with several strategies: early update, max-violation, full update (even though Huang et al. (2012) discourage its use), several variants of those (e.g. to optimize the similarity of the positive and negative configurations), together with the restart option and arbitrary exploration policies.

As explained in §2.6, local and global training are unified under the same framework (Algorithm 1), in which the basic training unit is the model update, and all training strategies follow the same workflow: initialize a beam in a given configuration, extend the beam repeatedly until an error is flagged, select a pair of update configurations among the candidates, perform the update, and iterate until the example is considered processed.

The various update strategies for global training can themselves be unified under the same framework, as shown by Algorithm 2 (Correct being the error criterion used by global dynamic oracles): with this viewpoint, EarlyUpdate and MaxViolation only differ in the choice of $\mathrm{i}_{\mathtt{update}}$, which is actually how they are implemented in

[3]Compared to their work, we enriched the representation of the stack and buffer pointers, with the current number of children, the two leftmost and the two rightmost children. This enables rich feature templates like Zhang and Nivre (2011)'s.

PanParser (with lazy expansion of the beam and anticipation of errors). Appendix B further describes how those strategies fit within the dynamic oracle framework.

Algorithm 2: Basic scheme for training strategies (Aufrant et al., 2016a).

At time t_0: $B_0 = \{c_1, c_2, ..., c_{k'}\}, k' \leq k$
k: maximum beam size (1 for local training)
λ_i: optional focus on early/late transitions (1 in all state-of-the-art strategies)
Function *TrainingUnit*(B_0, y, θ)
  $B_1, B_2, ..., B_N \leftarrow \textsc{Decode}(B_0, \theta^{t_0}, k)$ such that $\textsc{Final}(B_N)$
  if $\neg \textit{Correct}_y(\text{top}(B_N)|B_0)$ **then**
    $i_0 \leftarrow$ index of the first error detection (in B_{i_0})
    The algorithm chooses in turn:
    • using $\{B_i\}_i, i_0, \theta^{t_0}, \textsc{Correct}_y$: a positive configuration c^+ for each derivation length
    • using $\{B_i\}_i, i_0, \theta^{t_0}, \{c^+\}$: a negative configuration c^- for each derivation length
    • using $\{B_i\}_i, i_0, \theta^{t_0}, \{c^+\}, \{c^-\}$: a derivation length i_{update}
    $c^+_{i_{update}} = c_{empty} \circ t^+_0 \circ \cdots \circ t^+_{i_{update}}$
    $c^-_{i_{update}} = c_{empty} \circ t^-_0 \circ \cdots \circ t^-_{i_{update}}$
    The global update is effected, e.g. with the perceptron rule:
    • $\theta^{t_0+1} \leftarrow \theta^{t_0} + \sum_{i=0}^{i_{update}} \lambda_i \left[\phi(t^+_i)) - \phi(t^-_i)\right]$

On top of the errors flagged by the oracle, PanParser also accepts other non-standard stopping criteria, like forcing the beam to reinitialize every few actions (thus preventing updates on very distant configurations).

3.3. Enriched input: partial trees and constraints

Compared to traditional implementations, PanParser makes an extensive use of the dynamic oracle framework to leverage partial trees in several ways: it supports training on partially annotated sentences (which enables robustness to incomplete datasets, but also fine-grained subsampling), predicting under partial constraints (when the head, or at least the dependency direction, is already provided for a few tokens) and training under constraints (for better train-test consistency, and using features extracted from the constraints). It is also possible, using the corresponding transition systems, to learn to predict partial trees directly – in the PanParser framework, training and prediction unfold as usual even for such partial parsers.

The reason why dynamic oracles enable training on examples with partial annotations is that they make the updates error-driven: when no information is provided, the cost is simply under-estimated and no update occurs. This behavior holds even for more complex dynamic oracles, either global or non-arc-decomposable (see Ap-

pendix A). So, provided that the action costs are implemented appropriately (i.e. not assuming the existence of annotations), which is the case in PanParser, training on such data is possible by design. Fine-grained subsampling can then be entertained by on-the-fly deletion of some reference dependencies, before training on a given example; error-driven training takes care of exploiting the remaining dependencies.

As for constrained training and prediction, they rely on a straightforward action filtering, based on dynamic oracles: restricting the search space to parses compatible with these constraints simply consists in restricting the legal actions to those that have zero cost with respect to the constraints. This way, the constraint dependencies are naturally respected and included by the parser, which in fact produces a standard derivation, without any pre- or post-processing.

3.4. Parser ensembling

Two strategies for parser ensembling are implemented under the PanParser framework: parser cascading (Aufrant and Wisniewski, 2017), which consists in pipelining a series of partial parsers, and MST-based reparsing (Sagae and Lavie, 2006). Reparsing enables a token-level vote on the output of several parsers; when weighting the contribution of each parser, which PanParser allows, this strategy can for instance be used in cross-lingual parsing, to combine various sources (Rosa and Žabokrtský, 2015).

3.5. Support for other structured prediction tasks: PoS tagging

PanParser also has a built-in PoS tagger, based on the same structured prediction framework. It shows how this framework can be used for other structured prediction tasks than dependency parsing. This unification also paves the way to joint tagging and parsing with PanParser.

The structure and usage of the tagger are similar to PanParser, albeit simpler because it does not involve transition systems. The main difference is that at training time, the tagger also builds a tag dictionary of unambiguous words, with almost always the same tag (and enough occurrences) in the dataset, and at prediction time it tries looking up the tag in the dictionary, before defaulting to actual predictions.

3.6. Error analysis

In order to facilitate extensive error analysis, PanParser is shipped with a series of evaluation tools, both for computing overall accuracies and fine-grained statistics. Several built-in criteria (PoS tags, dependency length, direction, position in sentence, word frequency, etc.) can be used (and combined) to guide the analysis or narrow the results. Examples of the studies enabled by PanParser are presented in Supplementary B.

4. Experiments

The accuracy of PanParser is evaluated on the 73 treebanks of the Universal Dependencies 2.0 (Nivre et al., 2017a,b). Table 2 reports the average scores achieved with the main few settings, together with similar measures for three other open source parsers: MaltParser 1.9 (Nivre et al., 2006a), MSTParser 0.5.1 (McDonald et al., 2005) and UDPipe 1.1 (Straka and Straková, 2017).[4]

Our system appears competitive with the other parsers, all of them being outperformed by an 8-sized beam PanParser. Further comparison with UDPipe reveals that both systems are actually on par on large treebanks (more than 500 sentences), while PanParser outperforms all parsers by a large margin on small treebanks (less than 500 sentences).

As a side note, the gains achieved by PanParser when changing the training strategy and Root position also appear consistent with the literature (Goldberg and Nivre, 2012; Zhang and Nivre, 2012; Huang et al., 2012; Ballesteros and Nivre, 2013), which validates previous results in this new framework.

5. Conclusion and future work

We have presented PanParser, a transition-based dependency parser implemented with the concern of algorithmic variation completeness, intended both for practical uses and as a parsing research tool. It currently supports numerous options and customizations for several aspects of the parsing algorithms.

PanParser is, however, still a work in progress, and we already plan several extra developments. We intend to take a further step to customization completeness, by allowing to parse without dummy Root token, and to extract arbitrary user-defined atomic features. We will also add built-in transition systems that are not stack- and buffer-based: the Covington system, based on two lists and a buffer, and for which Gómez-Rodríguez and Fernández-González (2015) already derived a dynamic oracle; the SwapStandard system (using a stack and a list), which requires deriving new efficient dynamic oracles; and the EasyFirst system, based on a single list. Another planned extension is to add relaxed types of arc constraints, e.g. ambiguous constraints, and span constraints.

Finally, we will add support for stateful classifiers to add a stack-LSTM parser implementation, and allow arbitrary joint prediction, which should achieve full PanParser support for state-of-the-art systems like that of Swayamdipta et al. (2016).

[4] We use the default settings for MaltParser (ArcEager parser with a linear classifier and no pseudo-projectivization) and MSTParser (first-order projective parser), and Straka (2017)'s hyperparameters for UDPipe.

System	Root position	Greedy	Greedy dynamic	Early update	Max-violation
ArcEager	First	77.89	78.97	80.29	80.36
	Small || Large	66.27 || 81.15	68.17 || 82.00	68.48 || 83.60	68.42 || 83.71
	Last	78.63	79.43	80.35	80.40
	Small || Large	67.60 || 81.72	68.70 || 82.44	68.58 || 83.66	68.87 || 83.63
ArcHybrid	First	75.72	76.54	79.39	79.78
	Small || Large	66.56 || 78.29	66.49 || 79.36	66.72 || 82.95	68.43 || 82.96
	Last	76.02	77.05	79.70	79.86
	Small || Large	66.74 || 78.62	67.42 || 79.76	68.39 || 82.87	68.61 || 83.02
MaltParser				72.88	
				58.87 || 76.82	
MSTParser				79.52	
				65.59 || 83.43	
UDPipe				79.47	
				64.48 || 83.67	

Table 2: Average UAS achieved by PanParser on UD 2.0 with various strategies, compared to several open source parsers. 'Greedy' results are computed with a static oracle, but for fair comparison of both oracles the non-projective examples are also exploited (using a precomputed reference approximated by a dynamic oracle). 'Greedy dynamic' chooses exploration after each update. The 'Early update' and 'Max-violation' strategies use global dynamic oracles without restart. 'Small' and 'Large' results distinguish the treebanks under and over 500 sentences. For fair comparison, UDPipe is trained without pre-trained embeddings, which would have significantly increased the available information.

Acknowledgments

This work has been partly funded by the French *Direction générale de l'armement* and by the *Agence Nationale de la Recherche* (ParSiTi project, ANR-16-CE33-0021).

Bibliography

Agirre, Eneko, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. Improving Dependency Parsing with Semantic Classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 699–703, Portland, Oregon, USA, 6 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-2123`.

Alberti, Chris, David Weiss, Greg Coppola, and Slav Petrov. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing, pages 1354–1359, Lisbon, Portugal, 9 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1159.

Andor, Daniel, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, 8 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1231.

Aufrant, Lauriane and Guillaume Wisniewski. LIMSI@CoNLL'17: UD Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 163–173, Vancouver, Canada, 8 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K17-3017.

Aufrant, Lauriane, Guillaume Wisniewski, and François Yvon. Ne nous arrêtons pas en si bon chemin: améliorations de l'apprentissage global d'analyseurs en dépendances par transition. In *Actes de la 23e conférence sur le Traitement Automatique des Langues Naturelles*, pages 248–261, 2016a.

Aufrant, Lauriane, Guillaume Wisniewski, and François Yvon. Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 119–130, Osaka, Japan, 12 2016b. The COLING 2016 Organizing Committee. URL http://aclweb.org/anthology/C16-1012.

Aufrant, Lauriane, Guillaume Wisniewski, and François Yvon. Don't Stop Me Now! Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 318–323, Valencia, Spain, 4 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-2051.

Aufrant, Lauriane, Guillaume Wisniewski, and François Yvon. Exploiting Dynamic Oracles to Train Projective Dependency Parsers on Non-Projective Trees. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 413–419, New Orleans, Louisiana, 6 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N18-2066.

Ballesteros, Miguel and Bernd Bohnet. Automatic Feature Selection for Agenda-Based Dependency Parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 794–805, Dublin, Ireland, 8 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1076.

Ballesteros, Miguel and Joakim Nivre. MaltOptimizer: An Optimization Tool for MaltParser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–62, Avignon, France, 4 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E12-2012.

Ballesteros, Miguel and Joakim Nivre. Going to the Roots of Dependency Parsing. *Computational Linguistics*, 39(1):5–13, 2013. URL http://www.aclweb.org/anthology/J/J13/J13-1002.pdf.

Ballesteros, Miguel, Yoav Goldberg, Chris Dyer, and Noah A. Smith. Training with Exploration Improves a Greedy Stack LSTM Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010, Austin, Texas, 11 2016. Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1211.

Björkelund, Anders and Joakim Nivre. Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 76–86, Bilbao, Spain, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W15-2210.

Bohnet, Bernd, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428, 2013.

Chen, Danqi and Christopher Manning. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, 10 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1082.

Chen, Xinchi, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. Transition-based Dependency Parsing Using Two Heterogeneous Gated Recursive Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1879–1889, Lisbon, Portugal, 9 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1215.

Choi, Jinho D and Nicolas Nicolov. K-best, locally pruned, transition-based dependency parsing using robust risk minimization. *Recent Advances in Natural Language Processing V*, 309: 205–216, 2009.

Choi, Jinho D. and Martha Palmer. Getting the Most out of Transition-based Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 687–692, Portland, Oregon, USA, 6 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-2121.

Collins, Michael and Brian Roark. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, 7 2004. doi: 10.3115/1218955.1218970. URL http://www.aclweb.org/anthology/P04-1015.

Covington, Michael A. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102, 2001.

de Lhoneux, Miryam, Sara Stymne, and Joakim Nivre. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104, Pisa, Italy, 9 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-6314.

Dozat, Timothy and Christopher D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. *ICLR 2017*, 2017. URL http://arxiv.org/abs/1611.01734.

Dyer, Chris, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers), pages 334–343, Beijing, China, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1033.

Fernández-González, Daniel and Carlos Gómez-Rodríguez. Improving Transition-Based Dependency Parsing with Buffer Transitions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 308–319, Jeju Island, Korea, 7 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D12-1029.

Fernández-González, Daniel and Carlos Gómez-Rodríguez. A Full Non-Monotonic Transition System for Unrestricted Non-Projective Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 288–298, Vancouver, Canada, 7 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1027.

Fernández-González, Daniel and Carlos Gómez-Rodríguez. A Dynamic Oracle for Linear-Time 2-Planar Dependency Parsing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 386–392, New Orleans, Louisiana, 6 2018a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N18-2062.

Fernández-González, Daniel and Carlos Gómez-Rodríguez. Non-Projective Dependency Parsing with Non-Local Transitions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 693–700, New Orleans, Louisiana, 6 2018b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N18-2109.

Goldberg, Yoav and Michael Elhadad. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, 6 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1115.

Goldberg, Yoav and Joakim Nivre. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, 12 2012. The COLING 2012 Organizing Committee. URL http://www.aclweb.org/anthology/C12-1059.

Goldberg, Yoav and Joakim Nivre. Training Deterministic Parsers with Non-Deterministic Oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414, 2013. ISSN 2307-387X. URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/145.

Goldberg, Yoav, Kai Zhao, and Liang Huang. Efficient Implementation of Beam-Search Incremental Parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 628–633, Sofia, Bulgaria, 8 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-2111.

Goldberg, Yoav, Francesco Sartorio, and Giorgio Satta. A Tabular Method for Dynamic Oracles in Transition-Based Parsing. *Transactions of the Association for Computational Linguistics*, 2:119–130, 2014. ISSN 2307-387X. URL https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/302.

Gómez-Rodríguez, Carlos and Daniel Fernández-González. An Efficient Dynamic Oracle for Unrestricted Non-Projective Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–261, Beijing, China, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2042.

Gómez-Rodríguez, Carlos and Joakim Nivre. A Transition-Based Parser for 2-Planar Dependency Structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1492–1501, Uppsala, Sweden, 7 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1151.

Gómez-Rodríguez, Carlos and Joakim Nivre. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845, 2013.

Gómez-Rodríguez, Carlos, Francesco Sartorio, and Giorgio Satta. A Polynomial-Time Dynamic Oracle for Non-Projective Dependency Parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927, Doha, Qatar, 10 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1099.

Haulrich, Martin. Transition-Based Parsing with Confidence-Weighted Classification. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 55–60, Uppsala, Sweden, 7 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-3010.

Honnibal, Matthew and Mark Johnson. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, 9 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1162.

Honnibal, Matthew, Yoav Goldberg, and Mark Johnson. A Non-Monotonic Arc-Eager Transition System for Dependency Parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172, Sofia, Bulgaria, 8 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3518.

Huang, Liang and Kenji Sagae. Dynamic Programming for Linear-Time Incremental Parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden, 7 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1110.

Huang, Liang, Suphan Fayong, and Yang Guo. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, 6 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1015.

Kiperwasser, Eliyahu and Yoav Goldberg. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016. ISSN 2307-387X. URL https://transacl.org/ojs/index.php/tacl/article/view/885.

Kübler, Sandra, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009.

Kuhlmann, Marco, Carlos Gómez-Rodríguez, and Giorgio Satta. Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA, 6 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1068.

Lacroix, Ophélie, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. Frustratingly Easy Cross-Lingual Transfer for Transition-Based Dependency Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, 6 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1121.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, 10 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/H/H05/H05-1066.

Nilsson, Peter and Pierre Nugues. Automatic Discovery of Feature Sets for Dependency Parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 824–832, Beijing, China, 8 2010. Coling 2010 Organizing Committee. URL http://www.aclweb.org/anthology/C10-1093.

Nivre, Joakim. An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, IWPT 2003, Nancy, France, 2003.

Nivre, Joakim. Incrementality in Deterministic Dependency Parsing. In Keller, Frank, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain, 7 2004. Association for Computational Linguistics.

Nivre, Joakim. Algorithms for Deterministic Incremental Dependency Parsing. *Comput. Linguist.*, 34(4):513–553, 2008. ISSN 0891-2017. doi: 10.1162/coli.07-056-R1-07-027. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Nivre, Joakim. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, 8 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1040.

Nivre, Joakim and Daniel Fernández-González. Arc-eager parsing with the tree constraint. *Computational linguistics*, 40(2):259–267, 2014.

Nivre, Joakim, Johan Hall, and Jens Nilsson. Memory-Based Dependency Parsing. In Ng, Hwee Tou and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics.

Nivre, Joakim, Johan Hall, and Jens Nilsson. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, volume 6, pages 2216–2219, 2006a.

Nivre, Joakim, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City, 6 2006b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-2933.

Nivre, Joakim, Yoav Goldberg, and Ryan McDonald. Constrained arc-eager dependency parsing. *Computational Linguistics*, 40(2):249–527, 2014.

Nivre, Joakim, Željko Agić, Lars Ahrenberg, et al. Universal Dependencies 2.0, 2017a. URL http://hdl.handle.net/11234/1-1983. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague.

Nivre, Joakim, Željko Agić, Lars Ahrenberg, et al. Universal Dependencies 2.0 – CoNLL 2017 Shared Task Development and Test Data, 2017b. URL http://hdl.handle.net/11234/1-2184. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Pitler, Emily and Ryan McDonald. A Linear-Time Transition System for Crossing Interval Trees. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 662–671, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1068.

Qi, Peng and Christopher D. Manning. Arc-swift: A Novel Transition System for Dependency Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117, Vancouver, Canada, 7 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-2018.

Rosa, Rudolf and Zdeněk Žabokrtský. KLcpos3 - a Language Similarity Measure for Delexicalized Parser Transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2040.

Sagae, Kenji and Alon Lavie. Parser Combination by Reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, 6 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N06/N06-2033.

Stenetorp, Pontus. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*, 2013.

Straka, Milan. CoNLL 2017 Shared Task - UDPipe Baseline Models and Supplementary Materials, 2017. URL http://hdl.handle.net/11234/1-1990. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Straka, Milan and Jana Straková. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, 8 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K17-3009.

Swayamdipta, Swabha, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Greedy, Joint Syntactic-Semantic Parsing with Stack LSTMs. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 187–197, Berlin, Germany, 8 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K16-1019.

Weiss, David, Chris Alberti, Michael Collins, and Slav Petrov. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1032.

Zeman, Daniel and Philip Resnik. Cross-Language Parser Adaptation between Related Languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42, 2008.

Zhang, Yue and Stephen Clark. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, 10 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D08-1059.

Zhang, Yue and Joakim Nivre. Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, 6 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-2033.

Zhang, Yue and Joakim Nivre. Analyzing the Effect of Global Learning and Beam-Search on Transition-Based Dependency Parsing. In *Proceedings of COLING 2012: Posters*, pages 1391–1400, Mumbai, India, 12 2012. The COLING 2012 Organizing Committee. URL http://www.aclweb.org/anthology/C12-2136.

Zhao, Kai, James Cross, and Liang Huang. Optimal Incremental Parsing via Best-First Dynamic Programming. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 758–768, Seattle, Washington, USA, 10 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1071.

Zhou, Hao, Yue Zhang, Shujian Huang, and Jiajun Chen. A Neural Probabilistic Structured-Prediction Model for Transition-Based Dependency Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, 7 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1117.

Appendix A. Deriving a dynamic oracle

In this appendix, we describe more precisely the dynamic oracle framework and how to derive such oracles, as well as some alterations made to the original framework to extend its applicability. Indeed, PanParser does not implement the action cost, but rather a generalized version which simplifies the oracle derivation for some systems, and also enables straightforward approximations in cases when dynamic oracles have only been inexactly derived (as done by Aufrant et al. (2018)) or not derived at all. We discuss here the formal grounding of this extension, as well as its limits.

To specify dynamic oracles, Goldberg and Nivre (2012) formally define the cost of an action as '*the loss difference between the minimum loss tree reachable before and after*' performing the action in question. Considering the minimum loss is equivalent to computing the maximum UAS achieved on the given example, or (without normalization) the maximal number of correct attachments (henceforth denoted CA) on that sentence. So, the cost is evaluated by computing the maximum CA over all derivations resulting from the given configuration (c), and the maximum over only those following the given action (t).

$$\textsc{Cost}(c,t) = \left[\max_{t_1,\cdots,t_{end}} \mathrm{CA}(c \circ t_1 \circ \cdots \circ t_{end})\right] - \left[\max_{t_2,\cdots,t_{end}} \mathrm{CA}(c \circ t \circ t_2 \circ \cdots \circ t_{end})\right]$$

By definition of the maximum, Cost is always non-negative, and in every configuration at least one action has zero cost.

Arc-decomposable systems Exhaustively exploring all the successor derivations is computationally too expensive, and thus Goldberg and Nivre (2013) define the notion of arc-decomposition to simplify cost computation. A transition system is *arc-decomposable* if for any configuration c, all arcs reachable from c (i.e. predicted by at least one transition sequence after c) can be reached conjointly by a single transition sequence. This means that at the level of the transition system, there is no interaction between predicted arcs, and no incompatibility effect.

If we define $\textsc{ForbiddenArcs}(c,t)$ as the number of arcs that are reachable from c but not from $c \circ t$, Goldberg and Nivre (2013) state that for arc-decomposable systems, these arcs are the only source of cost, and thus:

$$\textsc{Cost}(c,t) = \textsc{ForbiddenArcs}(c,t)$$

Non-arc-decomposable systems When this property does not hold, on the other hand, there are extra sources of cost to account for, because of incompatible arcs. In case of such incompatibilities, at some point, adding a gold arc will indeed imply renouncing to another gold arc, thereby inserting an error. But this cost cannot be attributed to the given action, it is in fact due to a much earlier action, which introduced

the incompatibility. Besides, sometimes in such cases, ForbiddenArcs is non-zero for all legal actions, in which case it is obviously not identical to the Cost function.

There are two main strategies to compute the action cost in non-arc-decomposable systems. The first is to explicitly compute the loss before and after the action, typically using dynamic programming (Goldberg et al., 2014), and then retain the difference. The second is to directly model the cost, by formalizing the configurations holding arc incompatibilities, and detecting when such incompatibilities are inserted (Gómez-Rodríguez and Fernández-González, 2015). When possible, this is computationally cheaper than a full loss computation.

Relaxed action cost To formalize the cost in a non-arc-decomposable system, we define ExpectedCost(c) as the number of arcs that are still reachable from c but do not belong to the final output (considering some final configuration, reachable from c and with maximal UAS). This counts the number of current incompatibilities. The action cost then decomposes as:

$$\textsc{Cost}(c,t) = \textsc{ForbiddenArcs}(c,t) + (\textsc{ExpectedCost}(c \circ t) - \textsc{ExpectedCost}(c))$$

We now introduce the RelaxedCost function, defined as:

$$\textsc{RelaxedCost}(c,t) = \textsc{ForbiddenArcs}(c,t) + \textsc{ExpectedCost}(c \circ t)$$

from which ensues:

$$\begin{aligned} \textsc{Cost}(c,t) &= \textsc{RelaxedCost}(c,t) - \textsc{ExpectedCost}(c) \\ \textsc{ExpectedCost}(c) &= \textsc{RelaxedCost}(c,t) - \textsc{Cost}(c,t) \leq \textsc{RelaxedCost}(c,t) \end{aligned}$$

and because at least one action has zero cost:

$$\begin{aligned} \textsc{ExpectedCost}(c) &= \min_t \textsc{RelaxedCost}(c,t) \\ \textsc{RelaxedCost}(c,t) &= \textsc{ForbiddenArcs}(c,t) + \min_{t'} \textsc{RelaxedCost}(c \circ t, t') \\ \textsc{Cost}(c,t) &= \textsc{RelaxedCost}(c,t) - \min_{t'} \textsc{RelaxedCost}(c,t') \end{aligned}$$

In other words, the RelaxedCost function computes an overestimate of Cost, that repeatedly counts the cost of incompatibilities, as long as they are not resolved, and not only when they are introduced. Thus, it may happen that no action has a zero RelaxedCost, but the actual cost can be retrieved by shifting all costs by the minimum RelaxedCost, which corresponds to the current ExpectedCost. Hence, in this framework, the optimal actions are not those with zero cost but with minimal cost.

These definitions have notably two useful properties, which make the use of the alternate definition transparent. First, for arc-decomposable systems, ExpectedCost

is null, so RelaxedCost = Cost. Second, since $\min_t \text{Cost}(c,t) = 0$, in both cases (RelaxedCost and Cost), shifting by the minimum cost always yields Cost values.

Consequently, in PanParser, we define optimal actions as *minimum-cost actions*, and transition system implementations are supposed to compute either one of Cost (when computed as a loss difference) and RelaxedCost (when modeled explicitly).

In practice, defining the action cost explicitly then consists in listing as usual the arcs that the action makes unreachable, as well as the causes of arc incompatibilities in the future configuration.

Consequences of under-estimated costs Exhibiting all causes of incompatibilities is often a tedious task, it is even more so to *prove* that the list is exhaustive, as done by Gómez-Rodríguez and Fernández-González (2015) for the Covington system. We have not yet done this study for all non-arc-decomposable systems in PanParser, and have settled for now for firm beliefs: the non-arc-decomposable costs have indeed been tested against exhaustive search on all possible configurations, but for short sentences only.

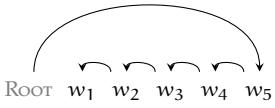
So, what happens if we have missed a given type of incompatibility? Or worse, if we miss all of them and simply use ForbiddenArcs for a non-arc-decomposable system? Using minimum-cost instead of zero-cost actions in fact strongly alleviates such issues.

Indeed, with an under-estimated cost, some actions introducing incompatibilities may be deemed correct, later resulting in configurations where all actions forbid some reference arc, even though no error has been detected in the past. With standard cost definition and a zero-cost criterion, this case is not covered, and training would presumably stop. But with the minimum-cost criterion there are always gold actions, whether the cost is correctly defined or not, and training can consequently go on transparently.

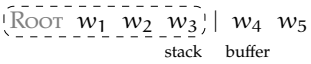
The only consequence on training is that the cost under-estimation introduces for the oracle a preference towards late resolution of inconsistencies: in case of two incompatible arcs, the parser will prefer actions that keep both options as long as possible over actions that forbid one of them right away. Figure 2 shows how bad this tendency can be. However, whether such cases have a strong impact on accuracy remains to be ascertained, on a per-case basis.

Consequently, the minimum-cost criterion makes it possible to use under-estimated costs, typically by ignoring non-arc-decomposability, but such unsound training has unknown consequences on accuracy. We thus advocate to empirically assess its effects for each considered system.

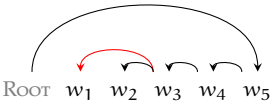
Non-projective examples Aufrant et al. (2018) have shown how dynamic oracles (with a minimum-cost criterion) make it possible to train a projective parser on non-projective sentences; this directly results from their ability to *accept* past errors and do



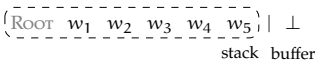
(a) Reference parse tree.



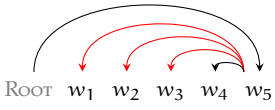
(b) Stack and buffer of the (already suboptimal) configuration to evaluate; all stack elements are unattached.



(c) Best tree, reached by Left-Left-Shift-Left-Shift-Left-Right.



(d) With under-estimated cost: preference for Shift-Shift, which do not forbid any arc.



(e) With under-estimated cost: best tree from there, reached by Left-Left-Left-Left-Right.

Figure 2: Consequences of training with an under-estimated cost ignoring arc incompatibilities, using ArcStandard with Root in first position.

their best to select good decisions anyway. The issue of non-projectivity is indeed exactly the same as that of arc incompatibilities: when two crossing edges are reachable, only one can actually belong to the final output.

Hence, from the oracle point of view, the initial empty configuration already comes with embedded 'past errors' (the incompatibilities due to edge crossings). As is the case for non-arc-decomposable systems, the cost incurred by these incompatibilities is not due to actions to come, but should be attributed to previous actions, taken in a fictive history before the initial configuration. As such, the natural behavior of dynamic oracles is to ignore this cost.

The costs of built-in transition systems have not been adapted yet to acknowledge arc incompatibilities due to non-projectivity. For now, we consequently use under-estimated costs for those sentences, which empirically remains better than discarding or projectivizing all non-projective sentences (Aufrant et al., 2018).

Systems without known dynamic oracle Defining the action cost is sufficient for using the transition system with any training option. However, for some transition systems, the cost of an action may be difficult to express, or computationally too expensive. In this case, it is still possible to define the cost as a degenerated version of a static oracle: the transition designated by the heuristics used for pre-computing references is given cost 0, all other transitions are given cost 1. This method ensures that any transition system can be incorporated in the PanParser, even though in this case it will not be fully compatible with other components (no dynamic oracle, so no error exploration and no constrained parsing).

Appendix B. Global dynamic oracles

In PanParser, the training procedure is mostly based on the concept of global dynamic oracles (Aufrant et al., 2017), which is a direct extension of usual dynamic oracles to global training.

Similarly to local dynamic oracles which deem incorrect the *actions* which introduce a new error into the final parse, global dynamic oracles deem incorrect the *transition sequences* which introduce a new error into the final parse. Hence, given an initial configuration (not necessarily empty or gold), the correct configurations are those from which the maximum UAS is the same as the initial maximum.

The Boolean function that tests this condition, denoted $\textsc{Correct}_y(c'|c)$, can thus be efficiently computed using the Cost function: a configuration c' is considered as Correct in the context of a configuration c, if there exists a sequence of transitions $t_1, \dots, t_n$ such that $c' = c \circ t_1 \circ \cdots \circ t_n$ and $\textsc{Cost}(c, t_1) = \textsc{Cost}(c \circ t_1, t_2) = \dots = \textsc{Cost}(c \circ \cdots \circ t_{n-1}, t_n) = 0$.

In other words, PanParser does not need to compute reference derivations explicitly, it just has to check the cost of each action it performs, and track the hypotheses that are still correct and those which are not.

Algorithm 3 (on the following page) shows how Correct is used to apply the early-update and max-violation strategies with dynamic oracles.

Algorithm 3: Global dynamic oracle: error criterion and choice of an update configuration pair.

c_0: configuration to start decoding from
$\text{top}_\theta(\cdot)$: best scoring element according to θ
NEXT(c): the set of all successors of c (or only c if it is final)
Function *FINDVIOLATION*(c_0, *y*, θ)
  Beam $\leftarrow \{c_0\}$
  while $\exists c \in$ *Beam*, $\neg$*FINAL*(c) **do**
    Succ $\leftarrow \cup_{c \in \text{Beam}} \text{NEXT}(c)$
    Beam $\leftarrow$ k-best(Succ, θ)
    if $\forall c \in$ *Beam*, $\neg$*CORRECT*$_y(c|c_0)$ **then**
      gold $\leftarrow \{c \in \text{Succ} | \text{CORRECT}_y(c|c_0)\}$
      return gold, Beam
  gold $\leftarrow \{c \in \text{Beam} | \text{CORRECT}_y(c|c_0)\}$
  return gold, Beam
Function *EARLYUPDATEORACLE*(c_0, y, θ)
  gold, Beam $\leftarrow$ FINDVIOLATION(c_0, y, θ)
  return $\text{top}_\theta(\text{gold})$, $\text{top}_\theta(\text{Beam})$
Function *MAXVIOLATIONORACLE*(c_0, y, θ)
  gold, Beam $\leftarrow$ FINDVIOLATION(c_0, y, θ)
  candidates $\leftarrow \{(\text{top}_\theta(\text{gold}), \text{top}_\theta(\text{Beam}))\}$
  while $\exists c \in$ *Beam*, $\neg$*FINAL*(c) **do**
    Succ $\leftarrow \cup_{c \in \text{Beam}} \text{NEXT}(c)$
    Beam $\leftarrow$ k-best(Succ, θ)
    $\text{Succ}^+ \leftarrow \cup_{c \in \text{gold}} \{c' \in \text{NEXT}(c) | \text{CORRECT}_y(c'|c_0)\}$
    gold $\leftarrow$ k-best(Succ^+, θ)
    candidates $\leftarrow$ candidates $+ (\text{top}_\theta(\text{gold}), \text{top}_\theta(\text{Beam}))$
  return $\arg\max_{c^+, c^- \in \text{candidates}} (\text{score}_\theta(c^-) - \text{score}_\theta(c^+))$

Address for correspondence:
Lauriane Aufrant
lauriane.aufrant@limsi.fr
LIMSI – 508 rue John von Neumann, 91405 Orsay, France

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 87–96

An Easily Extensible HMM Word Aligner

Jetic Gū,[a b] Anahita Mansouri Bigvand,[a] Anoop Sarkar[a]

[a] Simon Fraser University, Burnaby, Canada
[b] Zhejiang University, Hangzhou, China

Abstract

In this paper, we present a new word aligner with built-in support for alignment types, as well as comparisons between various models and existing aligner systems. It is an open source software that can be easily extended to use models of users' own design. We expect it to suffice the academics as well as scientists working in the industry to do word alignment, as well as experimenting on their own new models. Here in the present paper, the basic designs and structures will be introduced. Examples and demos of the system are also provided.

1. Introduction

Word alignment is a very important component in a machine translation system. Classical approaches include the IBM Models 1–5 (Brown et al., 1993) and the Hidden Markov Model (Vogel et al., 1996; Och and Ney, 2000a), which usually work quite well but insufficient for specific language pairs (Chinese-English for example, as shown in Section 4.1). As more human annotated data became available, a lot of supervised and semi-supervised algorithms were also proposed and had shown improvements, as demonstrated in Mansouri Bigvand et al. (2017).

In this paper we present a new open source HMM Aligner. The HMM Aligner (from this point forward will simply be referred to as The Aligner) not only contains built-in classic models, but also some of the supervised learning models as well (Section 3.3), while providing an extensible API. Users can easily combine existing models to form new learning sequences. For those that are familiar with the BaumWelch algorithm of HMM, it would only take a matter of minutes to implement their own extensions to the HMM model using The Aligner. The Aligner is written in Python using libraries such as Numpy so that it is efficient and also highly readable.

 Corresponding author: jeticg@sfu.ca
Cite as: Jetic Gū, Anahita Mansouri Bigvand, Anoop Sarkar. An Easily Extensible HMM Word Aligner. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 87–96. doi: 10.2478/pralin-2018-0008.

The comparisons of The Aligner and current systems will be provided in Section 2, while the design and models of The Aligner are in Section 3. In Section 4 experiment results are presented to demonstrate the capability of The Aligner.

Current version of The Aligner is available at `https://github.com/sfu-natlang/HMM-Aligner` under the MIT license.

2. Existing Systems

GIZA++ (Och and Ney, 2003) has long been the de facto software to use and compare to when doing word alignment. Although generally considered as reliable, it is highly sophisticated, making it very difficult to extend.

BerkeleyAligner (Liang et al., 2006) is a word aligner developed by The Berkeley NLP Group. Comparing to GIZA++ it is simpler to setup and use. It claims higher accuracy than GIZA++.

FastAlign (Dyer et al., 2013) is a very simple and fast word aligner. It claims better performance both in terms of speed and accuracy than GIZA++.

The aligners above are all made production ready, and there are not plenty of resources on how to build extensions. It would be relatively difficult if one wishes to develop their own HMM models using these software.

The presence of The Aligner aims at solving that exact problem. It provides an easily extendable interface, plenty of example models and it is written in a very friendly and flexible language: Python. Also it has built-in support for alignment type, which is a truly valuable resource in NLP with huge potential.

3. Design and Structure of The Aligner API

This section provides a brief introduction to how The Aligner works and an overview of its API design.

3.1. Workflow

The workflows of the training process and decoding process are presented in Figure 1.

In the beginning of a training sequence, the dataset will first be lexicalised, during which the original text in the dataset are converted into numbers using dictionary created with the training corpus. Each unique word from the source language and the target language will receive a unique index on which making it easier to do operations.

Then, during the training process the specified model will be loaded. Each model has its own training sequence, which performs the EM algorithm on different parts of the dataset in the order of ones choosing. The example in Figure 1 is the training sequence of HMM model, in which the translation probability is initialised using IBM model 1, and then the HMM model is trained. The details of this model will be introduced later in Section 3.3.

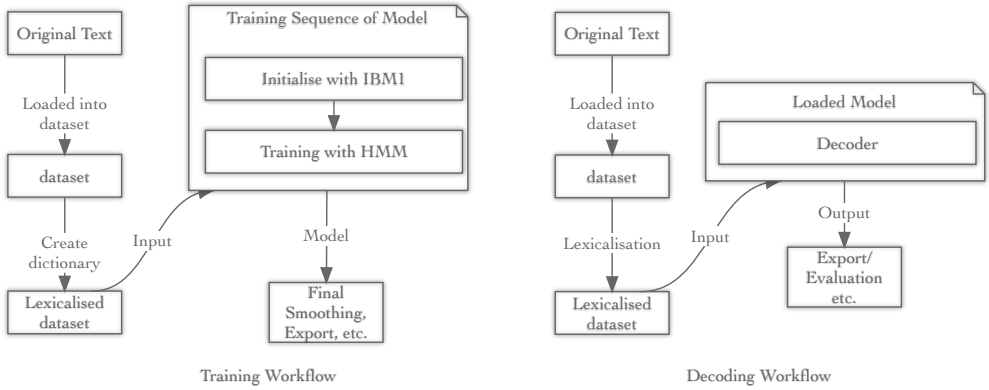


Figure 1. Workflow

During the evaluation process, lexicalised dataset is fed into the decoder to get the alignment.

3.2. Base Models

The Aligner provides two base models on which all other built-in models are built: the IBM model 1 (Brown et al., 1993) (Eq 1) and the HMM model (Och and Ney, 2000a) (Eq 2). Here $\mathbf{f}$ is the source sentence, and $\boldsymbol{e}$ the target sentence. $\mathbf{a}$ is the alignment from source to target: the jth source word is aligned to the a_jth target word. I is the length of the source sentence.

$$\Pr_{IBM1}(\mathbf{a}|\mathbf{f},\boldsymbol{e}) = \prod_{j=1}^{I} \mathbf{P}(f_j|e_{a_j}) \tag{1}$$

$$\mathbf{Pr}_{HMM}(\mathbf{a}|\mathbf{f},\boldsymbol{e}) = \prod_{j=1}^{I} [\mathbf{P}(a_j|a_{j-1},I)\mathbf{P}(f_j|e_{a_j})] \tag{2}$$

Both base models come with highly customisable API for training. Users using The Aligner can focus on modifying the models, applying smoothing, and designing their own training sequences on a higher level without having to worry about data structure and other lower level detail. All of the computational parts are optimised using Numpy and Cython to ensure good performance.

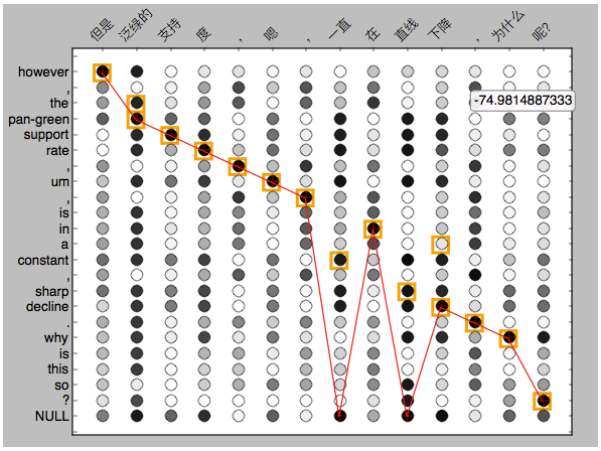


Figure 2. Screenshot of the figure of a pair of sentences. The gold alignment is represented with square boxes.

3.3. Alignment Type

Alignment of words between different languages can reflect more than just semantic meaning, it could also have been others including but not limited to function link, clausal link, modifier link, and also language specific links (Li et al., 2010). This information has proven to be quite useful for doing both supervised and semi-supervised aligner training as demonstrated in Mansouri Bigvand et al. (2017).

The Aligner provides built-in model that utilises alignment types (Eq 3) presented in Mansouri Bigvand et al. (2017). It uses both alignment type and POS tag information to enhance the baseline model to achieve better results. $\mathbf{h}$ represents the alignment type of the alignment: h_j is the alignment type of jth source word and a_jth target word.

$$\mathbf{Pr}_{\mathrm{HMMType}}(\mathbf{a}, \mathbf{h}|\mathbf{f}, \mathbf{e}) = \prod_{j=1}^{I} [\mathbf{P}(a_j|a_{j-1}, I)\mathbf{P}(f_j|e_{a_j})\mathbf{P}(h_j|f_j, e_{a_j})] \tag{3}$$

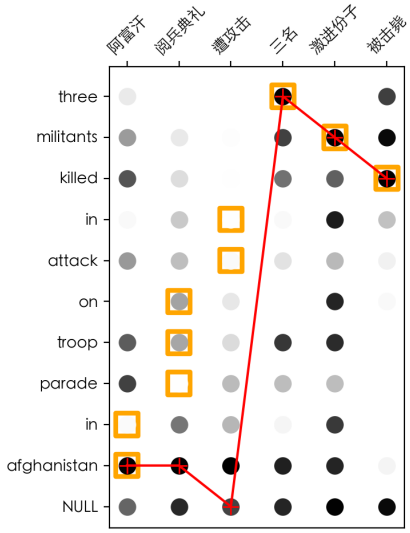


Figure 3. HMM Baseline

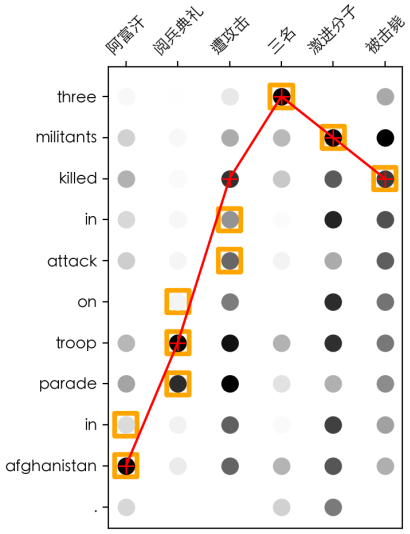


Figure 4. HMM With Alignment Type

3.4. Extended HMM models

Example implementation of extensions to the HMM model, as presented in K. Toutanova's 2002 paper is also provided. In the experiment section, we will be comparing the following extension (Eq 4. $\mathbf{P^*}(a_j|a_{j-1}, e_{a_{j-1}}, I)$ is defined in Eq 5):

$$\mathbf{Pr}_{\mathrm{HMMType}}(\mathbf{a}, \mathbf{h}|\mathbf{f}, \boldsymbol{e}) = \prod_{j=1}^{I}[\mathbf{P^*}(a_j|a_{j-1}, e_{a_{j-1}}, I)\mathbf{P}(f_j|e_{a_j})\mathbf{P}(\mathrm{fTag}_j|\mathrm{eTag}_{a_j})], \quad (4)$$

$$\mathbf{P^*}(a_j|a_{j-1}, e_{a_{j-1}}, I) = \delta(a_j, a_{j-1})\mathbf{P}(\mathbf{stay}|e_{a_{j-1}}) + \left\{ \begin{array}{r} (1-\delta(a_j, a_{j-1})) \\ (1-\mathbf{P}(\mathbf{stay}|e_{a_{j-1}})) \\ (\mathbf{P}(a_j, a_{j-1}, I)) \end{array} \right\} \quad (5)$$

$\delta(a_j, a_{j-1})$ in Eq 5 is the Kronecker delta function. $\mathbf{P}(\mathbf{stay}|e_{a_{j-1}})$ is the probability of the alignment of the next source word being the same target word, given the aligned target word of the previous source word.

3.5. Displaying Alignment Scores

The Aligner also includes built-in feature of displaying the alignment score of each pair of words of sentences during decoding (Figure 2). The score of each pair is presented with grey-scaled colours. In any column, if the colours of two circles appear close then the score of these two are very close, v.v.. If one moves the cursor on any circle, the score of that corresponding pair of words will appear next to the cursor, making it easier to do debugging and present results. The gold alignment is also displayed to make doing comparisons easier.

This feature is very useful for debugging and also comparing results of different models.

3.6. Loading And Saving Models

The Aligner allows users to save the trained models for reuse. Not only are built-in models savable but also user customised models as well. It does not require any extra code to make it work, the API will automatically clean up the data structures, make saving and loading the least of ones concern when it comes to model design.

3.7. Intersection

Intersecting the alignment results from source-target training and target-source training usually gives better results, as demonstrated by Liang et al. (2006). The Aligner has built-in support for such intersection for all models without the need of extra code. Also, since intersection is done in parallel, it will not require extra time to train and decode.

4. Experiments

The experiments will focus on the quality of alignments produced by The Aligner's built-in models, including the models with alignment types and POS tags, by comparing Alignment Error Rates (Och and Ney, 2000b) and F1 Scores.

The datasets used in the experiments are: first for the datasets with human annotated alignment types and POS tags, we used the GALE Chinese-English Parallel Aligned Treebank[1] which includes parallel text from news broadcasts, news articles, and online discussion panels; then, experiments were also carried out on the French-English Hansard Canadian Parliament dataset[2] and the German-English Dataset from the European Parliament Proceedings Parallel Corpus (Koehn, 2005). The latter two datasets are all parliament proceedings. Sentences in parliament proceedings appear

[1]Catalog numbers: LDC2014T25; LDC2015T04; LDC2015T06; LDC2015T18; LDC2016T19; LDC2017T05

[2]Catalog numbers: LDC95T20

more formal than daily conversations, are highly structured and sophisticated, and contain significant amount of legal-domain words. They also do not come with alignment types, so we only compared the quality of models that does not require alignment types, including the extended HMM model.

The German-English dataset and French-English dataset we used are all parliament proceedings, which are also highly popular dataset for NLP experiments. Although all in spoken language, the sentences often are highly structured and sophisticated, with significant amount of legal-domain words.

POS tags of datasets that originally do not contain POS tags are obtained using The Stanford POS Tagger (Toutanova et al., 2003).

Table 1 gives information on the sizes of the datasets. Table 2 contains information regarding percentages of words according to their occurrences in each dataset.

language pair	Unique Words		POS Tags		Sentences	
	source	target	source	target	train	test
Chinese-English	107 503	49 557	40	57	125 989	1956
French-English	92 280	82 074	18	45	951 462	447
German-English	395 952	133 835	27	55	1 890 489	150

Table 1. Information on Datasets

occurrences	Chinese-English		French-English		German-English	
	source	target	source	target	source	target
=1	48.00%	32.08%	35.77%	35.49%	50.32%	41.32%
≤3	70.18%	54.86%	55.98%	57.93%	70.16%	60.44%
≤5	78.15%	63.94%	64.01%	63.93%	76.95%	67.50%
≤10	86.20%	74.13%	72.90%	72.86%	83.98%	75.25%

Table 2. Percentage of words according to their occurrence

4.1. Experiments on Datasets with Alignment Types and POS Tags

Table 3 shows the performance of built-in models mentioned in Section 3.2 and also the result of Fast Aligner, GIZA++, and Berkeley Aligner on Chinese-English dataset. For all models, experiments were run on the same settings. It is apparent that models supporting alignment types and POS tags produce alignments with better quality.

Model	Precision	Recall	AER	F1-score
IBM1	0.5279	0.4204	0.5320	0.4680
IBM1 (Intersect)	0.8457	0.3686	0.4866	0.5134
HMM	0.7233	0.5063	0.4044	0.5956
HMM (Intersect)	0.9135	0.4431	0.4032	0.5968
HMM+Extensions	0.6865	0.5465	0.3915	0.6085
HMM+Extensions (Intersect)	0.8907	0.4758	0.3798	0.6202
HMM+Type	0.7257	**0.6189**	0.3319	0.6681
HMM+Type (Intersect)	0.9154	0.5690	**0.2982**	**0.7018**
GIZA++(Model 4)	0.6513	0.5825	0.3850	0.6150
GIZA++ (Manual Intersect)	**0.9481**	0.4049	0.4325	0.5675
Fast-Align	0.6263	0.6142	0.3798	0.6202
Fast-Align (Manual Intersect)	0.8674	0.5064	0.3605	0.6395
Berkeley-Aligner	0.7638	0.6116	0.3207	0.6793

Table 3. Chinese-English Dataset test results

4.2. Experiments on Datasets without Alignment Types

Experiments in this section are carried out on the French-English dataset (Table 4) and German-English dataset (Table 5).

Model	Precision	Recall	AER	F1-score
IBM1	0.5570	0.7038	0.3928	0.6218
HMM	0.7930	0.8462	0.1877	0.8187
HMM+Extension	0.7663	0.8834	0.1936	0.8207
HMM+Extension (Intersect)	0.9389	0.7979	0.1264	0.8627
GIZA++ (Model 4)	0.7848	0.7940	0.2115	0.7894
GIZA++ (Manual Intersect)	**0.9773**	0.7214	0.1548	0.8301
Fast-Align	0.7679	0.8294	0.2091	0.7975
Fast-Align (Manual Intersect)	0.7584	0.8826	0.1997	0.8158
Berkeley-Aligner	0.8677	**0.9113**	**0.1151**	**0.8899**

Table 4. French-English Dataset test results

Model	Precision	Recall	AER	F1-score
IBM1	0.5982	0.6388	0.3830	0.6178
HMM	0.7253	0.7513	0.2626	0.7381
HMM+Extension	0.7333	0.7693	0.2500	0.7509
HMM+Extension (Intersect)	0.9295	0.7066	0.1941	0.8029
GIZA++ (Model 4)	0.8611	0.7429	0.2006	0.7976
GIZA++ (Manual Intersect)	**0.9593**	0.6349	0.2334	0.7641
Fast-Align	0.7275	0.7587	0.2581	0.7428
Fast-Align (Manual Intersect)	0.8718	0.6240	0.2694	0.7274
Berkeley-Aligner	0.8898	**0.8227**	**0.1435**	**0.8549**

Table 5. German-English Dataset test results

As shown by the data, The Aligner performs fairly well despite its simplicity. It is able to produce competitive if not better results than the other Aligners. Of course these experiments do not represent the full potential of the built-in models. By applying various smoothing techniques for each individual language pair one can surely achieve better results. It is not this paper's intention to declare that The Aligner is a far better alternative to the other software available, but only that it is highly competitive while having the unique advantage of being easy to extend.

5. Conclusions

This project is maintained by the Simon Fraser University Natural Language Lab. We have plans to gradually include more sample models in the future.

The Aligner aims at providing an easy-to-extend interface to users wishing to do word alignment. The Aligner can also be used for educational purposes such as teaching students to do word alignment and learning HMM models and alignments. With its simplicity and flexibility, we believe it will be proven to be very useful for researchers as well as industrial productions.

Bibliography

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017. URL `http://dl.acm.org/citation.cfm?id=972470.972474`.

Dyer, Chris, Victor Chahuneau, and Noah. A Smith. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the Conference of NAACL: Human Language Technologies*, ACL 2013, pages 644–649. ACL, 2013. URL `http://repository.cmu.edu/cgi/viewcontent.cgi?article=1038&context=lti`.

Koehn, Philipp. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT. URL http://mt-archive.info/MTS-2005-Koehn.pdf.

Li, Xuansong, Niyu Ge, Stephen Grimes, Stephanie M. Strassel, and Kazuaki Maeda. Enriching word alignment with linguistic tags. In *In Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010.

Liang, Percy, Ben Taskar, and Dan Klein. Alignment by Agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA, 2006. ACL. doi: 10.3115/1220835.1220849. URL http://dx.doi.org/10.3115/1220835.1220849.

Mansouri Bigvand, Anahita, Te Bu, and Anoop Sarkar. Joint prediction of word alignment with alignment types. In *Transactions of ACL*, TACL 2017. ACL, 2017.

Och, Franz Josef and Hermann Ney. A Comparison of Alignment Models for Statistical Machine Translation. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 1086–1090, Stroudsburg, PA, USA, 2000a. ACL. doi: 10.3115/992730.992810. URL http://dx.doi.org/10.3115/992730.992810.

Och, Franz Josef and Hermann Ney. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of ACL*, pages 440–447, 2000b.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

Toutanova, Kristina, 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 NAACL on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. ACL. doi: 10.3115/1073445.1073478. URL http://dx.doi.org/10.3115/1073445.1073478.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. HMM-based Word Alignment in Statistical Translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. ACL. doi: 10.3115/993268.993313. URL http://dx.doi.org/10.3115/993268.993313.

Address for correspondence:
Jetic Gū
jeticg@sfu.ca
NATLANG Lab, Simon Fraser University
8888 University Dr, Burnaby
BC V5A1S6, Canada

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 97–112

A Probabilistic Approach to Error Detection&Correction for Tree-Mapping Grammars

Tim vor der Brück

School of Information Technology, Lucerne University of Applied Sciences and Arts, Switzerland

Abstract

Rule-based natural language generation denotes the process of converting a semantic input structure into a surface representation by means of a grammar. In the following, we assume that this grammar is handcrafted and not automatically created for instance by a deep neural network. Such a grammar might comprise of a large set of rules. A single error in these rules can already have a large impact on the quality of the generated sentences, potentially causing even a complete failure of the entire generation process. Searching for errors in these rules can be quite tedious and time-consuming due to potentially complex and recursive dependencies. This work proposes a statistical approach to recognizing errors and providing suggestions for correcting certain kinds of errors by cross-checking the grammar with the semantic input structure. The basic assumption is the correctness of the latter, which is usually a valid hypothesis due to the fact that these input structures are often automatically created.

Our evaluation reveals that in many cases an automatic error detection and correction is indeed possible.

1. Introduction

In NLG, one common task is to transform a set of nested feature-value pairs into a constituency tree structure by means of a set of grammar rules. These grammatical rules are often context free production rules enriched by context-sensitive constraints. Figure 1 illustrates the assumed generation model.

While implementing the grammar, a grammar developer will probably commit errors. On the one hand, such an error can be conceptual, i.e., the developer did not take something into account that is crucial for the functioning of the generation process. Such errors can easily require a major redesign of the grammar. On the

 Corresponding author: tim.vorderbrueck@hslu.ch
Cite as: Tim vor der Brück. A Probabilistic Approach to Error Detection&Correction for Tree-Mapping Grammars. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 97–112. doi: 10.2478/pralin-2018-0009.

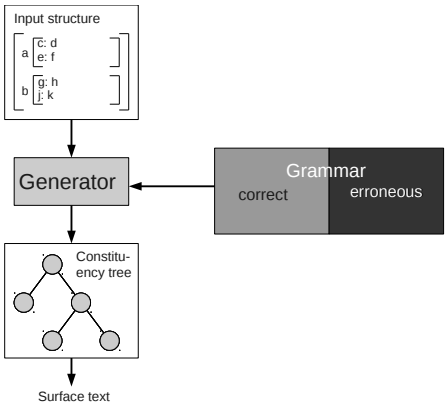


Figure 1: Generation Model.

other hand, an error can be a simple oversight, for instance, the grammar developer misspelled a path expression or a category name. The latter type of errors are the ones we will focus here.

The effect of a grammar error can be manifold. In case the grammar developer accidentally omitted an RHS (right hand side) of a rule, the generated text will probably be incomplete. If he instead switched two RHS, then the output will most likely be scrambled. In both of these cases, the developer will usually get a good hunch regarding the nature and location of the error already by looking at the generated text. However, if the grammar developer accidentally selected the incorrect category or path expression, then in many cases the generation process will fail completely. Such errors are particularly hard to trace. A further source of difficulty for error analysis is the use of recursion, which can result in a deeply nested constituency tree.

In this work, we propose a statistical approach to automatically detecting incorrect path expression and category selections and to providing suggestions for correcting these kinds of errors. It is based on the static grammar verification method introduced by vor der Brück and Busemann (2006) and contains comprehensive and considerable extensions, such as a more detailed evaluation, an in-depth description of a statistical disambiguation method, and discussions of false alarms. The term "*path expression*" is coined to access parts of the input structure. This is quite similar to the term "*relative XPath expression*" in XML. The hypothesis stated in this work is that the semantics contained in the available input structures contains sufficient information to detect those errors.

Our method is intended to be used together with template based text generation systems, in which the grammar rules access an externally defined input structure. Examples of such systems are TG/2 (Busemann, 1996), XtraGen (Stenzhorn, 2003)

and D2S (Theune et al., 2001). It is implemented as plugin for the eGram grammar workbench (Busemann, 2004). eGram supports the comfortable development of text generation grammars for the formalisms TG/2 and XtraGen. Furthermore, eGram is seamlessly integrated with the TG/2/XtraGen generation component, which makes it possible to view the result of the generation process directly inside the environment. The most part of a grammatical rule definition using eGram is acomplished by selecting the appropriate path variables (path variables, see page 104), access functions and categories from several comboboxes.

While errors concerning the rule syntax are practically impossible, there are additional types of errors that cannot occur when writing the grammar with a simple text editor. For instance, it is easily possible that the grammar developer clicks on the category or path expression that is displayed below or above the category / path expression that was actually intended. A typical error that can occur when either using eGram or a text editor is that the grammar developer copies a rule, modifies it afterward to build a new one and the modification is incomplete. This type of error is also possible with eGram since this editor provides a similar functionality.

2. Related Work

Gardent and Narayan (2012) as well as vor der Brück and Stenzhorn (2008) describe a dynamic method that identifies errors in generation grammars by running a generator on semantic input structures. A static approach, however, as proposed here is usually much faster. For instance, our system can do several grammar verification iterations in a few seconds. Furthermore, a static approach does not suffer from potential endless recursion preventing a termination of a dynamic error analysis. For a distinction between static and dynamic approaches see Daich et al. (1994).

There is also some prior work conducted on automated error detection for parsing (the opposite operation to generation). Kok et al. (2009) and van Noord (2004) present an approach, where a large corpus is parsed by an analyzer and n-grams leading to a parsing failure are marked as suspicious. These suspicious n-grams can then be used to track down errors in the tokenizer, lexicon and grammatical rules.

Checking linguistic grammars is closely related to program code analysis since a generator is nothing else than a certain type of software. Zeller (2005) proposed a dynamic testing algorithm to determine the causes of program crashes (segmentation faults). This algorithm isolates the error by subsequently executing different parts of the computer program with varying program states, which is called delta debugging.

3. Input Structure and Grammar Formalism

An input structure is a semantic representation of a certain domain. We assume that it can be structured as nested name-value pairs. The primitive value can either be a string or a number. The concatenation of two input structures forms a new input; the

value part of an input structure can again be an input structure. Formally, we define an input structure as follows.

Definition 1. *An input structure* **InStruc** *defines the semantics of a domain, assigns names to components within the domain, and records values of these components, where components at the same granularity shall be named differently.*

$$\begin{aligned} \mathbf{InStruc} &::= \mathtt{string}|\mathtt{number} && (1)\\ \mathbf{InStruc} &::= (\mathtt{name}, \mathbf{InStruc}) && (2)\\ \mathbf{InStruc} &::= \mathbf{InStruc}; \mathbf{InStruc} && (3) \end{aligned}$$

An input structure is a labeled tree structure. Its edges are labeled with names; its nodes are input structures; its leaves can either be strings or numbers. Children of the same node are labeled with different names.

For example, the following input structure describes the semantics of a temporal duration. The corresponding labeled tree structure is illustrated in Formula 4. Each leaf node in the tree is uniquely identified by the sequence of labels starting from the root node. For example, the sequence "*from, hour*" uniquely identifies the leaf node 10. Generally, given a start node, a label sequence defines a travel from this start node. A travel is a selection of nodes among the descendant nodes.

$$\left[\begin{array}{ll} \mathtt{from} & \left[\begin{array}{ll} \mathtt{hour}: & 10\\ \mathtt{min}: & 20\\ \mathtt{sec}: & 30 \end{array}\right]\\ \mathtt{to} & \left[\begin{array}{ll} \mathtt{hour}: & 12\\ \mathtt{min}: & 30\\ \mathtt{sec}: & 10 \end{array}\right] \end{array}\right] \tag{4}$$

Definition 2. *Given an input structure* **InStruc**, *a sequence of strings* $a_1, \ldots, a_n$ *is a* **path expression** *of* **InStruc**, *if and only if, for any* i *there is an input structure* $\mathbf{InStruc}_i$, *such that* $(a_i, \mathbf{InStruc}_i)$ *is within* **InStruc**. *A* **path expression** $a_1, \ldots, a_n$ *is written as* $[a_1/\ldots/a_n]$. *The selection process is defined as follows.*

$$[a_1/\ldots/a_n] \circ \mathbf{InStruc} = \begin{cases} [a_2/\ldots/a_n] \circ \mathbf{InStruc}_{a_1}, & (a_1, \mathbf{InStruc}, \mathbf{InStruc}_{a_1})\\ \emptyset & \text{otherwise} \end{cases} \tag{5}$$

where $(a_1, \mathbf{InStruc}, \mathbf{InStruc}_{a_1})$ *means that there is an edge labeled with* a_1 *between* **InStruc** *and* $\mathbf{InStruc}_{a_1}$.

The uniqueness of the selection process is guaranteed by the structure of the input. For the convenience of computation, we define the empty path expression $[\epsilon]$ as follow.

$$[\epsilon] \circ \mathbf{InStruc} = \mathbf{InStruc} \tag{6}$$

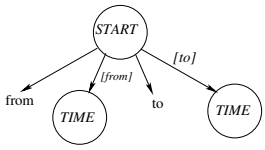


Figure 2: Tree representation of a grammar rule that generates a time interval.

Our task is to transform a given input structure into a linguistic surface structure. In the former example, we need to transform the semantic representation of the temporal duration into the phrase "*from 10:20:30 to 12:30:10*". To this end, we employ grammatical rules.

The grammar considered here follows the TG/2 formalism (Busemann, 1996, 2005), which is also used by the XtraGen generator (Stenzhorn, 2003). TG/2 is basically a context-free grammar formalism consisting of production rules. Each production rule consists of one LHS (left hand side) category and several RHS (right hand side) categories or functions or simple surface text strings. The RHS categories and functions are associated to path expressions that specify the part of the input structure accessible for such categories / functions.

$$\begin{aligned} &\mathtt{R}_1 : \mathit{START} \rightarrow \text{ from } \mathit{TIME}[\mathit{from}] \text{ to } \mathit{TIME}[\mathit{to}] \\ &\mathtt{R}_2 : \mathit{TIME} \rightarrow \mathit{toString}([\mathtt{hour}]) : \mathit{toString}([\mathtt{min}]) : \mathit{toString}([\mathtt{sec}]) \end{aligned} \tag{7}$$

START is the top-level category the generation process kicks off with. "*toString*" is a function that generates the part of the input structure that is referenced by the associated path expression. These rules are used to display a time expression in a formatted way. A tree representation of rule $\mathtt{R}_1$ is given in Figure 2.

After applying rule $\mathtt{R}_1$ on the category *START* with the input structure depicted in formula 4, an incomplete constituency tree is created with two preliminary leaf nodes labeled with *TIME*. Rule $\mathtt{R}_2$ is then applied afterward on the *TIME* nodes and can only access the part of the input structure selected by the path expression [*from*] (or [*to*] respectively). If converted into an absolute path expression, the path expression "*hour*", located on the right side of rule $\mathtt{R}_2$, would evaluate to [*/from/hour*] (or [*/to/hour*] respectively). The generated surface string for the given input structure is: "*from 10:20:30 to 12:30:10*".

4. Error Identification with Left and Right Attributes

We define parent-child relationship for labels.

Definition 3. *A label* $\mathtt{a}$ *is defined to be a parent label of label* $\mathtt{b}$*, if* $\mathtt{a}$ *is the label of an input structure* **InStruc***, such that* $\mathtt{b}$ *is a label within* **InStruc***.*

Next, we define left and right grammar attributes of categories. If an RHS of a rule with LHS category C is labeled with the path expression $[a_1, \ldots, a_n]$, the path component a_1 is added to the set of right attributes of category C. If a path is empty, then the right grammar attributes of the associated RHS category are inherited by the LHS category C. Similarly, left grammar attributes of a category are defined as the last path components belonging to a rule transition edge leading to this category. If the path is empty, then the left grammar attributes of the LHS category are inherited. The right attributes give a characterization of a category. Consider, for example, the categories *START* and *TIME* and the two rules from the last section:

R_1 : *START* $\rightarrow$ from *TIME*[*from*] to *TIME*[*to*].

R_2 : *TIME* $\rightarrow$ *toString*([*hour*]) : *toString*([*min*]) : *toString*([*sec*])

The right grammar attributes of *START* are "*from*" and "*to*", the right grammar attributes of *TIME* are "*hour*", "*min*", "*sec*".

In addition, we define the right and left validation attributes of categories. Validation attributes build a superset of all allowed grammar attributes and are extracted from both the input structure and the grammar. Consider the case that there exists an RHS labeled with a path "*pe*" leading to category C. Let "*e*" be the last component of path "*pe*". Then all possible child elements in the input structures of "*e*" belong to the right validation attributes of C. This is the case for all RHS of rules leading to category C. If "*pe*" is the empty path expression, then the right validation attributes of the LHS category are added to C. If C is the START (top-level) category, then all top-level input structure attributes are the right validation attributes of C. Formally, the right validation attributes can be specified as follows (vor der Brück and Busemann, 2006):

$$r_C := \begin{cases} \{d | \exists R \in \mathit{Rules}, \\ \mathit{pe} \in \mathit{PE}, D \in \mathit{Cat} : \\ R : D \rightarrow C[\mathit{pe}] \wedge \\ ((\mathtt{parent}(\mathit{pe}[|\mathit{pe}|], d) \vee \\ (\mathit{pe} = \varepsilon \wedge d \in r_D)\}, & C \neq \mathtt{START} \\ \mathtt{top}, & \mathtt{otherwise} \end{cases} \tag{8}$$

where

- *Rules*: set of all rules
- *Cat*: set of all categories
- *PE*: set of all path expressions
- *parent*(a, b): a relation that is fulfilled if and only if a occurs as parent of b in any of the input structures
- *top*: all elements occurring at the top-level of any input structure

The left validation attributes of C are defined similarly. Let C be the LHS category of a rule with one RHS labeled with path "*pe*". Let us consider the first element of

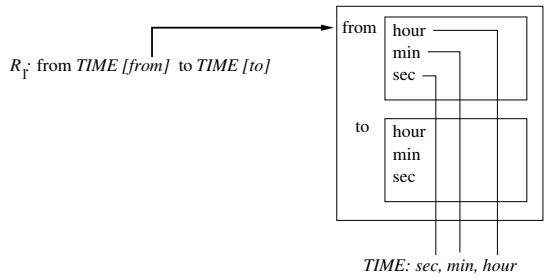


Figure 3: Construction of right validation attributes.

this path called "f". Then all possible parents of "f" in the input structures belong to the left validation attributes of C. If "*pe*" is the empty path, then all left validation attributes of the RHS category are added to the LHS category. This is the case for any such RHS. The process of extracting right and left validation attributes is illustrated in Figures 3, 4, and in Table 2. Formally, the left validation attributes of C are given as:

$$\begin{aligned} \mathrm{l_C} := \{ \mathrm{d} | \exists pe \in PE, \mathrm{R} \in Rules : \\ \mathrm{R} : \mathrm{C} \rightarrow \mathrm{D}[pe] \wedge \\ (\mathtt{parent}(\mathrm{c}, pe[1]) \vee \\ pe = \varepsilon \wedge \mathrm{d} \in \mathrm{l_D}) \end{aligned} \tag{9}$$

where D is either a category, a string-valued function or a string.

In order for a rule to be applicable with the current input structures, the right grammar attributes of its LHS (RHS) category must be contained in the right validation attributes of its LHS (RHS) category. Similarly, the left grammar attributes must be contained in the left validation attributes. To handle multiple errors, right grammar attribute mismatches are identified top-down (beginning with the START category). In this way, the right validation attributes that were introduced due to incorrect RHS paths can be removed. Analogously, left grammar mismatches are identified bottom up.

Now let us consider an example error of the grammar developer. We assume that he wanted to enter rules R_1 and R_2 but made a mistake at rule R_2.

$$\mathrm{R}_{2,\mathtt{error}} : TIME \rightarrow toString([\mathrm{to}]) : toString([\mathtt{min}]) : toString([\mathrm{sec}])$$

The right validation attributes of *TIME* do not change, but the right grammar attributes of *TIME* now contain the attribute "*to*", which is not contained in the right validation attributes of *TIME* and, therefore indicate an error in this RHS. The correct path must begin with one of the right validation attributes of *TIME*, i.e., "*hour*",

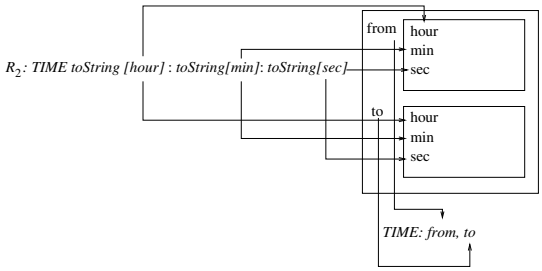


Figure 4: Construction of left validation attributes.

Cat	Left	Right
TIME	from,to	to,min,sec
START	-	from,to

Table 1: Grammar attributes derived by the rules R_1 and $R_{2,\mathtt{error}}$.

Cat	Left	Right
TIME	from,to	hour,min,sec
START	-	from,to

Table 2: Validation attributes derived by the rules R_1 and $R_{2,\mathtt{error}}$.

"*min*" or "*sec*". Table 1 shows the grammar and Table 2 the validation attributes of the categories. Since the selection of *min* or *sec* would lead to the same word generated multiple times (called double generation in short), which is quite unlikely in practice, hour can be correctly selected. Now consider a different error. Let us assume that the grammar developer wrote

$$R_{1,\mathtt{error}} : \mathit{START} \rightarrow \text{from } \mathit{TIME}[\mathit{min}] \text{ to } \mathit{TIME}[\mathit{to}] \tag{10}$$

instead of the correct rule R_1. The right validation attributes of the *START* category are not affected by this error and are the top-level input structure attributes "*to*" and "*from*". Thus, the attribute "*min*" is not contained in the right validation attributes of category *START* and is therefore detected as erroneous. The correct path expression must be either *[to]* or *[from]*. Again, the path *[to]* would lead to a double generation. Therefore, the correct path expression must be *[from]*. Note that in general an arbitrary number of path expressions can be created by combining attributes by path separators. In practice, these expressions are reduced to a finite set by the fact that the grammar editor requires all used path expressions to be associated to path variables. So only all existing path variable values have to be checked. In the example above, we employed a disambiguation heuristic to find a unique solution. In practice, there are many cases where the correct solution cannot so easily be found. Thus, in addition to

the heuristic "*double generation*", a statistical heuristic, which is explained in Section 6 in more detail, was used.

5. False Alarms

The errors detected by the method described here are only guaranteed to be actual errors (under the precondition that the input structures are always correct) if the empty path expression is never used in a grammar rule. It is actually possible that a false alarm is produced if a category can be reached from two different parent categories and one of the transitions is connected to the empty path expression. Consider for example the following grammar and input structure, which might not follow good design principles, but still leads to a successful generation of "Text2 Text1".

$$
\begin{aligned}
Q_1 &: \mathit{START} \rightarrow B[\epsilon] \\
Q_2 &: \mathit{START} \rightarrow C[\epsilon] \\
Q_3 &: B \rightarrow D[\epsilon] \\
Q_4 &: C \rightarrow D[a] \\
Q_4 &: D \rightarrow \mathit{toString}([c]) \\
Q_5 &: D \rightarrow \mathit{toString}([d]) \\
\text{Input} &: \begin{bmatrix} a & [d\ \text{“}\mathit{Text1}\text{“}] \\ c & \text{“}\mathit{Text2}\text{“} \end{bmatrix}
\end{aligned}
$$

B inherits among others the right grammar attribute d from category D, which is not contained in B's right validation attributes (a and c). One solution for this problem is to add all right validation attributes of C to category B or more generally: If a category C_1 is connected by the empty path expression with its child category C_2, all right validation attributes of all other categories leading to C_2 are added to C_1[1]. A similar approach can be employed for left attributes. The possible rule applications are visualized in Figure 5.

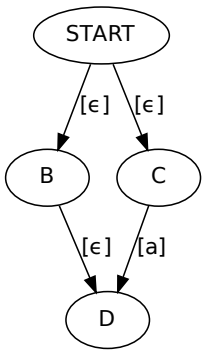


Figure 5: Graph that visualizes the rules where a false alarm is produced by the error-recognition algorithm.

[1]Note that this feature is not yet implemented in our current system.

6. Statistical approach to automatic error correction

Suppose that the error could be successfully spotted, but several path expressions are possible for correcting the erroneous RHS and the disambiguation methods mentioned above still leave several potential candidates. Now the statistical disambiguation method comes into play.

This method estimates, how probably a certain rule candidate is given all existing rules (the input structures are disregarded) and suggests the most probable one(s). Actually, the path expression '*pe*' is chosen that maximizes the posterior probability that the path expression '*pe*' occurs, given a transition from category LHS C_1 to RHS C_2, which can be formalized as follows:

- $pa : E \rightarrow P$: a function that assigns an edge $e \in E$ to a path expression $pe \in P$ (for the definition of a rule edge see Figure 2)
- $s : E \rightarrow Cat$: source category of an edge
- $d : E \rightarrow Cat$: destination category of an edge
- *Cat*: set of categories, *E*: set of edges

$$
\begin{aligned}
pe' &= \operatorname*{argmax}_{pe} P' \text{ with} \\
P' &= P(pa(e) = pe | s(e) = C_1 \wedge d(e) = C_2) \\
&\quad \text{(Theorem of Bayes)} \\
&= P(pa(e) = pe) \cdot \\
&\quad \frac{P(s(e) = C_1 \wedge d(e) = C_2 | pa(e) = pe)}{P(s(e) = C_1 \wedge d(e) = C_2)} \\
&\quad (P(s(e) = C_1 \wedge d(e) = C_2) \text{ is independent of } pe) \\
pe' &= \arg\max_{pe} P'' \\
P'' &= P(pa(e) = pe) \cdot \\
&\quad P(s(e) = C_1 \wedge d(e) = C_2 | pa(e) = pe) \\
&\quad \text{(Now we make the assumption that} \\
&\quad s(e) = C_1 \text{ and } d(e) = C_2 \\
&\quad \text{are approximately conditionally independent} \\
&\quad \text{given } pa(e) = pe. \\
&\quad \text{This assumption is made} \\
&\quad \text{to handle the sparse data problem} \\
&\quad \text{that usually shows up in} \\
&\quad \text{hand-written grammars.)}
\end{aligned}
$$

$$
\begin{aligned}
&\approx \quad P(pa(e) = pe)P(s(e) = C_1|pa(e) = pe) \cdot \\
&\qquad P(d(e) = C_2|pa(e) = pe) \\
&\qquad \text{(applying Theorem of Bayes again)} \\
&= \quad P(pa(e) = pe) \cdot \\
&\qquad P(pa(e) = pe|s(e) = C_1) \cdot \\
&\qquad P(pa(e) = pe|d(e) = C_2) \cdot \\
&\qquad \frac{P(s(e) = C_1)P(d(e) = C_2)}{P^2(pa(e) = pe)} \\
&= \quad P(pa(e) = pe|s(e) = C_1) \cdot \\
&\qquad P(pa(e) = pe|d(e) = C_2) \cdot \\
&\qquad \frac{P(s(e) = C_1)P(d(e) = C_2)}{P(pa(e) = pe)} \\
&\qquad (P(s(e) = C_1) \text{ and } P(d(e) = C_2 \\
&\qquad \text{are independent of } pe) \\
pe' &\approx \quad \arg\max_{pe} P''' \\
P''' &= \quad P(pa(e) = pe|s(e) = C_1) \cdot \\
&\qquad \frac{P(pa(e) = pe|d(e) = C_2)}{P(pa(e) = pe)}
\end{aligned}
$$

If there exists no RHS category due to the fact that the RHS is a function, then we instead determine the path expression pe with

$$\underset{pe}{\text{argmax}} P(pa(e) = pe|s(e) = C_1). \tag{11}$$

The most probably path expression(s) as obtained above are then suggested as correction. As usual, the probabilities are estimated by relative frequencies.

Let us now consider an example for this procedure. We extend the original grammar (rules R_1 and R_2, see page 102) by the following rules:

$$
\begin{aligned}
&R_3 : \mathit{TIME} \rightarrow \mathit{toString}([\mathit{hour}]) : \mathit{toString}([\mathit{min}]) \\
&R_4 : \mathit{TIME} \rightarrow \mathit{toString}([\mathit{hour}]) \\
&R_5 : \mathit{START} \rightarrow \text{from } \mathit{toString}([\mathit{from/loc}]) \text{ to} \\
&\qquad \mathit{toString}([\mathit{from/loc}])
\end{aligned} \tag{12}
$$

and add a second input structure:

$$
\begin{bmatrix} \mathit{from}: & [\mathit{loc} : \mathit{Bonn}] \\ \mathit{to}: & [\mathit{loc} : \mathit{Cologne}] \end{bmatrix} \tag{13}
$$

Let us now consider the case that rule R_2 was entered erroneously in the following way. Instead of the correct rule

$$R_2 : \ TIME \rightarrow toString[hour] : \\ toString[min] : toString[sec] \tag{14}$$

the grammar developer accidentally entered the incorrect rule:

$$R_{2,err} : TIME \rightarrow toString[from] : \\ toString[min] : toString[sec] \tag{15}$$

The analysis with validation and grammar attributes results in the fact that the incorrect path expression [*from*] has to be replaced by either [loc], [hour], [min], or [sec]. [min] and [sec] can be ruled out by the double generation rule so the alternatives [*from*] and [*loc*] remain possible. Now the probabilistic rule comes into play. The probability for [*loc*] is given by:

$$P(pa(e) = [loc] | \, \mathrm{source}(e) = TIME) = 0.0 \tag{16}$$

and the probability for [*hour*] is given by:

$$P(pa(e) = [hour] | \, \mathrm{source}(e) = TIME) = 1/3 \tag{17}$$

since there are 6 rule RHS with LHS *TIME* and 2 of them are associated to the path expression [*hour*]. Thus, the path expression [*hour*] is selected. Another heuristic is name comparison. Often, the paths and associated RHS categories have similar names. Therefore, we prefer paths that contain common substrings with the LHS/RHS categories. Finally, we use a heuristic that the empty path should not be suggested for LHS category, for which the set of right validation attributes is not empty, since presumably such a category matches a non-leaf node in the input structure.

Note that in some cases the generation fails because the path might be correct but the LHS (or RHS) category of some rule might be incorrectly chosen. In this case, instead of looking for the correct path we have to look for the category that best fulfills the validation attribute constraints of the erroneous rule.

7. Evaluation

For the evaluation, we used a grammar generating natural language descriptions of houses comprising of 267 rules, 96 categories and 104 different path variables. This grammar is assumed to be correct (errors displayed already for the unmodified grammar are ignored in the evaluation).

In general, there are two possibilities how to conduct the evaluation. The intrinsic approach is to look directly at the grammar and determine how many of the incorrect categories or path expression can be corrected. In contrast, we could evaluate

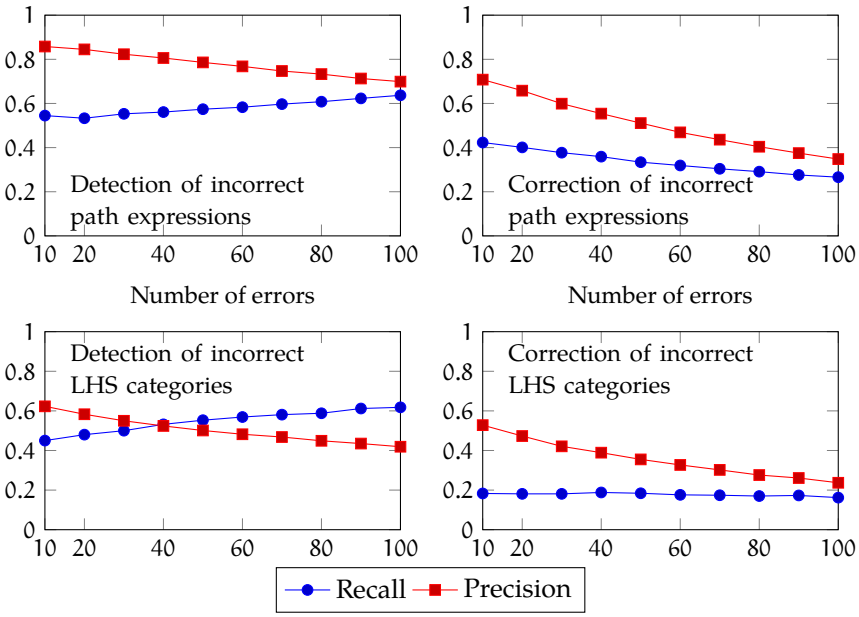


Figure 6: Recall and precision of detection (left) and correction (right) of incorrect path expressions (top) and LHS categories (bottom).

	Detection		Correction	
Errors	Prec.	Recall	Prec.	Recall
10	0.545	0.858	0.423	0.708
20	0.533	0.845	0.401	0.658
30	0.553	0.823	0.377	0.599
40	0.561	0.806	0.359	0.554
50	0.574	0.786	0.334	0.511
60	0.583	0.768	0.319	0.469
70	0.597	0.747	0.304	0.436
80	0.608	0.733	0.291	0.404
90	0.623	0.713	0.276	0.375
100	0.637	0.699	0.266	0.348

Table 3: Precision and recall for the correction of invalid path expressions.

	Detection		Correction	
Errors	Prec.	Recall	Prec.	Recall
10	0.450	0.623	0.183	0.528
20	0.480	0.583	0.181	0.473
30	0.500	0.550	0.181	0.421
40	0.532	0.524	0.188	0.389
50	0.553	0.501	0.184	0.355
60	0.569	0.482	0.176	0.327
70	0.581	0.468	0.174	0.302
80	0.588	0.449	0.170	0.276
90	0.612	0.435	0.173	0.261
100	0.618	0.419	0.162	0.237

Table 4: Precision and recall for the correction of invalid LHS categories.

our approach also extrinsically by comparing the actually generated text with the expected output and calculate some typical performance score like BLUE/METEOR or ROGUE. However, the extrinsic evaluation is not really adequate in this scenario, since most of the errors covered here would result not in a scrambled output but in no output at all. Thus, by employing an extrinsic evaluation, it would usually not be possible to discern for example, whether one or two errors were resolved. Likewise, it could normally not be decided, whether our method failed completely or was at least able to spot the error but then suggested the wrong correction. Hence, we opted to evaluate our approach intrinsically only. In particular, we insert errors randomly assuming a uniform error distribution by either choosing an RHS and modifying the path expression (path errors) or by selecting an LHS and changing its category (LHS category errors). In practice, a uniform distribution of the errors is rather unlikely. For instance, we would expect path expressions that are located nearby the correct one in the GUI list box to be chosen more often than path expressions that are far away. Unfortunately, it is very hard to obtain real error data. The number of inserted errors are varied from 10 to 100 in steps of 10. For each number of errors, this procedure is repeated one thousand times. See Figure 6 and Tables 3 and 4, for recall and precision of detection and correction of incorrect path expressions and LHS categories.

The precision of the error detection approach is defined by the quotient of the number of correctly determined errors and the number of total errors displayed. The percentual number of errors, which were detected correctly is called the recall of the error detection. An error is considered as detected if the incorrect rule and RHS (LHS in case of category errors) are recognized correctly.

The precision of the error correction is defined by the quotient of the number of the correct error-correction suggestions divided by the number of all suggestions. The percentual number of cases where the correct suggestions were found is called the recall of the error correction. For the evaluation of the error correction, we only regard the cases where the error was correctly detected. Note that there usually exist a lot of possible modifications that would make the grammar correct. However, for practical reasons, we only considered such a modification correct if it is exactly the inverse operation of the conducted modification. The evaluation showed that the erroneous RHS could be identified with an average precision of 54.5% (for 10 randomly inserted errors), which is far above the random baseline of $<1/267$ (analogously for the correction of rules), which means that our hypothesis that the input structures contain enough information to detect errors automatically cannot be rejected (significance level: 1%). The most difficult to detect are errors involving the empty path expressions, which can introduce a lot of ambiguities. Moreover, the recall degrades with an increasing number of errors, which is caused by the fact that left and right validation attributes become less reliable for error detection if the grammar contains a lot of errors. In contrast, the precision of our error detection approach is increasing with the number of errors since the proportion of erroneous rules become larger and therefore the a priori probability that a selected rule is erroneous increases.

For comparison, we converted the grammar into the XtraGen format and applied the method of (vor der Brück and Stenzhorn, 2008). Unfortunately, we did not get any result after several hours, which might be caused by an endless recursion in the generator, and aborted the run. A comparison with the approach of Gardent and Narayan (Gardent and Narayan, 2012) is not directly possible, since the authors focus especially on inputs in form of dependency trees and employ a generator based on tree adjoining grammars. However, in our method we assume a context free generation process, while tree adjoining grammars are actually context sensitive. Additionally, we do not make any assumption about the nature of our input despite being hierarchically ordered. In particular, the input structure of the grammar used for this evaluation constitutes no dependency tree.

8. Conclusion

A generation grammar might contain errors that result in an empty output for a given input structure. An empty output gives, in contrast to an ill-formed output, almost no clues about the reason for the generation failure. We presented and evaluated a method to detect and propose possible corrections for such errors automatically. The evaluation showed that in many cases a detection and correction was indeed possible.

For future work, we plan to investigate how to decide if a path expression or a category is actually incorrect. Also, we plan to recognize errors in RHS categories as well.

Currently, our approach is only used for checking text generation grammars. However, it could also be used to verify arbitrary XSLT stylesheets containing XPath expressions. Instead for categories we would then extract right and left attributes for stylesheet rules. The contents of the match attribute of an XML template would contribute both to the left grammar attributes as well as to the right validation attributes.

A completely self-correcting XSLT stylesheet or text generation grammar is still out of reach, but nevertheless, some ideas and concepts are shown how this goal can become a reality.

Acknowledgments

Hereby I thank the DFKI for its support of this work, especially for granting me free access to the eGram grammar workbench. Special thanks go to the associate head of their natural language processing group, Prof. Dr. Stephan Busemann.

Bibliography

Busemann, S. Best-first surface realization. In *Eight International Natural Language Generation Workshop*, pages 101–110, Brighton, England, 1996.

Busemann, S. eGram - A Grammar Development Environment and Its Usage for Language Generation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal, 2004. URL `http://www.dfki.de/dfkibib/publications/docs/busemann-LREC04.pdf`.

Busemann, S. Ten Years After: An Update on TG/2 (and Friends). In Wilcock, Graham, Kristiina Jokinen, Chris Mellish, and Ehud Reiter, editors, *Proceedings of the Tenth European Natural Language Generation Workshop (ENLG 2005)*, pages 32–39, Aberdeen, UK, 2005.

Daich, G., G. Price, B. Raglund, and M. Dawood. Software Test Technologies Report, 1994. URL `http://citeseer.ist.psu.edu/daich94software.html`.

Gardent, Claire and Shashi Narayan. Error Mining in Dependency Trees. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, South Korea, 2012.

Kok, Daniël, Jianqiang Ma, and Gertjan van Noord. A generalized method for iterative error mining in parsing results. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks (GEAF)*, Suntec, Singapore, 2009.

Stenzhorn, H. XtraGen. A natural language generation system using Java and XML technologies. Master's thesis, Saarland University, Department for Computational Linguistics, 2003.

Theune, M., Esther Klabbers, Jan Odijk, J.R. De Pijper, and Emiel Krahmer. From Data to Speech: A General Approach. *Natural Language Engineering*, 7(1), 2001.

van Noord, Gertjan. Error Mining for Wide-Coverage Grammar Engineering. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, Barcelona, Spain, 2004.

vor der Brück, Tim and Stephan Busemann. Automatic Error Correction for Tree-Mapping Grammars. In *Proceedings of KONVENS 2006*, pages 1–8, Konstanz, Germany, 2006. ISBN 3-89318-050-8.

vor der Brück, Tim and Holger Stenzhorn. A Dynamic Approach for Automatic Error Detection in Generation Grammars. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, Patras, Greece, 2008.

Zeller, A. Locating Causes of Program Failures. In *27th International Conference on Software Engineering (ICSE)*, Saint Louis, Missouri, USA, 2005.

Address for correspondence:
Tim vor der Brück
`tim.vorderbrueck@hslu.ch`
Suurstoffi 41b, CH 6343 Rotkreuz

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 113–124

NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning

Álvaro Peris, Francisco Casacuberta

Pattern Recognition and Human Language Technology Research Center, Universitat Politècnica de València, Spain

Abstract

We present NMT-Keras, a flexible toolkit for training deep learning models, which puts a particular emphasis on the development of advanced applications of neural machine translation systems, such as interactive-predictive translation protocols and long-term adaptation of the translation system via continuous learning. NMT-Keras is based on an extended version of the popular Keras library, and it runs on Theano and TensorFlow. State-of-the-art neural machine translation models are deployed and used following the high-level framework provided by Keras. Given its high modularity and flexibility, it also has been extended to tackle different problems, such as image and video captioning, sentence classification and visual question answering.

1. Introduction

To easily develop new deep learning models is a key feature in this fast-moving field. We introduce NMT-Keras[1], a flexible toolkit for neural machine translation (NMT), based on the Keras library for deep learning (Chollet et al., 2015). Keras is an API written in Python which provides high-level interfaces to numerical computation libraries, such as Theano (Theano Development Team, 2016) or TensorFlow (Abadi et al., 2016). Keras is well-structured and documented, with designing principles that make it modular and extensible, being easy to construct complex models.

Following the spirit of the Keras library, we developed NMT-Keras, released under MIT license, that aims to provide a highly-modular and extensible framework

[1] `https://github.com/lvapeab/nmt-keras`

 Corresponding author: `lvapeab@prhlt.upv.es`
Cite as: Álvaro Peris, Francisco Casacuberta. NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 113–124.
doi: 10.2478/pralin-2018-0010.

to NMT. NMT-Keras supports advanced features, including support of interactive-predictive NMT (INMT) (Barrachina et al., 2009; Peris et al., 2017c) protocols, continuous adaptation (Peris et al., 2017a) and active learning (Peris and Casacuberta, 2018b) strategies. An additional goal, is to ease the usage of the library, but allowing the user to configure most of the options involving the NMT process.

Since its introduction (Sutskever et al., 2014; Cho et al., 2014), NMT has advanced by leaps and bounds. Several toolkits currently offer fully-fledged NMT systems. Among them, we can highlight OpenNMT (Klein et al., 2017), Tensor2Tensor (Vaswani et al., 2018) or Nematus (Sennrich et al., 2017). NMT-Keras differentiates from them by offering interactive-predictive and long-term learning functionalities, with the final goal of fostering a more productive usage of the NMT system.

This document describes the main design and features of NMT-Keras. First, we review the deep learning framework which is the basis of the toolkit. Next, we summarize the principal features and functionality of the library. We conclude by showing translation and INMT results.

2. Design

NMT-Keras relies on two main libraries: a modified version of Keras[2], which provides the framework for training the neural models; and a wrapper around it, named Multimodal Keras Wrapper[3], designed to ease the usage of Keras and the management of data.These tools represent a general deep learning framework, able to tackle different problems, involving several data modalities. The problem of NMT is an instantiation of the sequence-to-sequence task, applied to textual inputs and outputs. NMT-Keras is designed to tackle this particular task. The reason for relying on a fork of Keras is because this allows us to independently design functions for our problems at hand, which may be confusing for the general audience of Keras. However, in a near future we hope to integrate our contributions into the main Keras repository. In this section, we describe these components and their relationships in our framework.

2.1. Keras

As mentioned in Section 1, Keras is a high-level deep learning API, which provides a neat interface to numerical computation libraries. Keras allows to easily implement complex deep learning models by defining the layers as building blocks. This simplicity, together with the quality of its code, has made Keras to be one of the most popular deep learning frameworks. It is also well-documented, and supported by a large community, which fosters its usage.

[2] https://github.com/MarcBS/keras

[3] https://github.com/lvapeab/multimodal_keras_wrapper

In Keras, a model is defined as a directed graph of layers or operations, containing one or more inputs and one or more outputs. Once a model has been defined, it is compiled for training, aiming to minimize a loss function. The optimization process is carried out, via stochastic gradient descent (SGD), by means of an optimizer.

Once the model is compiled, we feed it with data, training it as desired. Once a model is trained, it is ready to be applied on new input data.

2.2. Multimodal Keras Wrapper

The Multimodal Keras Wrapper allows to handle the training and application of complex Keras models, data management (including multimodal data) and application of additional callbacks during training. The wrapper defines two main objects and includes a number of extra features:

Dataset: A Dataset object is a database adapted for Keras, which acts as data provider. It manages the data splits (training, validation, test). It accepts several data types, including text, images, videos and categorical labels. In the case of text, the Dataset object builds the vocabularies, loads and encodes text into a numerical representation and also decodes the outputs of the network into natural language. In the case of images or videos, it also normalizes and equalizes the images; and can apply data augmentation.

Model wrapper: This is the core of the wrapper around Keras. It connects the Keras library with the Dataset object and manages the functions for training and applying the Keras models. When dealing with sequence-to-sequence models, it implements a beam search procedure. It also contains a training visualization module and ready-to-use convolutional neural networks (CNN) architectures.

Extra: Additional functionalities include extra callbacks, I/O management and evaluation of the system outputs. For computing the translation quality metrics of the models, we use the coco-caption evaluation tool (Chen et al., 2015), which provides common evaluation metrics: BLEU, METEOR, CIDEr, and ROUGE-L. Moreover, we modified it[4] to also include TER.

2.3. NMT-Keras

The NMT-Keras library makes use of the aforementioned libraries, for building a complete NMT toolkit. The library is compatible with Python 2 and 3. The training of NMT models is done with the `main.py` file. The hyperparameters are set via a configuration file (`config.py`), and can also be set from the command line interface. To train a NMT system with NMT-Keras is straightforward:

1. Set the desired configuration in `config.py`.
2. Launch `main.py`.

[4] `https://github.com/lvapeab/coco-caption`

The training process will then prepare the data, constructing the Dataset object and the Keras model. The default models implemented in NMT-Keras are an attentional RNN encoder–decoder (Bahdanau et al., 2015; Sennrich et al., 2017), and the Transformer model (Vaswani et al., 2017). Once the model is compiled, the training process starts, following the specified configuration. For translating new text with a trained model, we use beam search.

3. Features

As we keep our Keras fork constantly up-to-date with the original library, NMT-Keras has access to the full Keras functionality, including (but not limited to):

Layers: Fully-connected layers, CNN, recurrent neural networks (RNN), including long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), gated recurrent units (GRU) (Cho et al., 2014) and their bidirectional (Schuster and Paliwal, 1997) version, embeddings, noise, dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015) layers.

Initializers: The weights of a model can be initialized to a constant value, to values drawn from statistical distributions or according to the strategies proposed by Glorot and Bengio (2010), He et al. (2015) and Klambauer et al. (2017).

Optimizers: A number of SGD variants are implemented: vanilla SGD, RMSProp (Tieleman and Hinton, 2012), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Adam and Adamax (Kingma and Ba, 2014). The learning rate can be scheduled according to several strategies (linear, exponential, "noam" (Vaswani et al., 2017)).

Regularizers and constraints: Keras allows to set penalties and constraints to the parameters and to the layer outputs of the model.

Our version of Keras implements additional layers, useful for sequence-to-sequence problems:

Improved RNNs: All RNN architectures can be used in an autoregressive way, i.e. taking into account the previously generated token. They also integrate attention mechanisms, supporting the *add* (Bahdanau et al., 2015) and *dot* (Luong et al., 2015) models.

Conditional RNNs: Conditional (Sennrich et al., 2017) LSTM/GRU layers, consisting in cascaded applications of LSTM/GRU cells, with attention models in between.

Multi-input RNNs: LSTM/GRU networks with two and three different inputs and independent attention mechanisms (Bolaños et al., 2018).

Transformer layers: Multi-head attention layers, positional encodings and position-wise feed-forward networks (Vaswani et al., 2017).

Convolutional layers: Class activation maps (Zhou et al., 2016).

Finally, NMT-Keras supports a number of additional options. Here we list the most important ones, but we refer the reader to the library documentation[5] for an exhaustive listing of all available options:

Deep models: Deep residual RNN layers, deep output layers (Pascanu et al., 2014) and deep fully-connected layers for initializing the state of the RNN decoder.

Embeddings: Incorporation of pre-trained embeddings in the NMT model and embedding scaling options.

Regularization strategies: Label smoothing (Szegedy et al., 2015), early-stop, weight decay, doubly stochastic attention regularizer (Xu et al., 2015) and a fine-grained application of dropout.

Search options: Normalization of the beam search process by length and coverage penalty. The search can be also constrained according to the length of the input/output sequences.

Unknown word replacement: Replace unknown words according to the attention model (Jean et al., 2015). The replacement may rely on a statistical dictionary.

Tokenizing options: Including full support to byte-pair-encoding (Sennrich et al., 2016).

Integration with other tools: Support for Spearmint (Gelbart et al., 2914), for Bayesian optimization of the hyperparameters and Tensorboard, the visualization tool of TensorFlow.

Apart from these model options, NMT-Keras also contains scripts for ensemble decoding and generation of N-best lists; sentence scoring, model averaging and construction of statistical dictionaries (for unknown words replacement). It also contains a client-server architecture, which allows to access to NMT-Keras via web. Finally, in order to introduce newcomers to NMT-Keras, a set of tutorials are available, explaining step-by-step how to construct a NMT system.

3.1. Interactive-predictive machine translation

The interactive-predictive machine translation (IMT) framework constitutes an efficient alternative to the regular post-editing of machine translation; with the aim of obtaining high-quality translations minimizing the human effort required for this (Barrachina et al., 2009). IMT is a collaborative symbiosis between human and system, consisting in an iterative process in which, at each iteration, the user introduces a correction to the system hypothesis. The system takes into account the correction and provides an alternative hypothesis, considering the feedback from the user. Figure 1 shows an example of the IMT protocol.

[5] `https://nmt-keras.readthedocs.io`

This protocol has show to be especially effective when combined with NMT (Knowles and Koehn, 2016; Peris et al., 2017c). NMT-Keras implements the interactive protocols described by Peris et al. (2017c). Moreover, they can be combined with online learning (OL) or active learning methods, which allow to specialize the system into a given domain or to the preferences of a user (Peris and Casacuberta, 2018a). These protocols are also implemented in NMT-Keras. We built a demo website[6] of these interactive, adaptive systems using the client-server features of the toolkit (Figure 2).

Source:		They are lost forever .
Target:		Ils sont perdus à jamais .
0	**MT**	Ils sont perdus pour toujours .
1	**User**	*Ils sont perdus* à pour toujours .
	MT	*Ils sont perdus à* jamais .
2	**User**	*Ils sont perdus à jamais .*

Figure 1. IMT session. The user introduces in iteration 1 a character correction (boxed). The MT system modifies its hypothesis, taking into account this feedback.

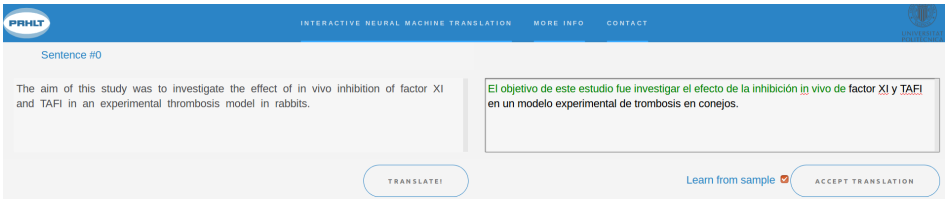


Figure 2. Frontend of the INMT with OL website built with NMT-Keras. In the left-side, the source sentence to translate is displayed. The system computes a translation hypothesis, located at the right frame. The user makes changes on this hypothesis. As a key is pressed, the hypothesis is immediately updated, taking into account the user correction. In this interaction protocol, the system validates the hypothesis prefix (validated in green), although we expect to support more flexible protocols (Peris et al., 2017c) in a future. This process is repeated until the desired translation is reached. Then, the user presses the Accept translation button, validating the translation. The system will use this sample to update the model if the "Learn from sample" checkbox is activated. The demo is available at: `http://casmacat.prhlt.upv.es/inmt`.

[6] `http://casmacat.prhlt.upv.es/inmt`.

3.2. Tackling other tasks

The modular design of Keras and Multimodal Keras Wrapper allows to use them to tackle other problems, following the spirit of NMT-Keras. These libraries have been used to perform video captioning (Peris et al., 2016), egocentric video captioning considering temporal contexts (Bolaños et al., 2018), text classification (Peris et al., 2017b), food recognition and localization (Bolaños and Radeva, 2016; Bolaños et al., 2017) and visual question answering (Bolaños et al., 2017).

4. Results

We report now results of NMT-Keras, assessing translation quality and INMT effectiveness. Due to space restrictions, we report results on two tasks: EU (Barrachina et al., 2009) and Europarl (Koehn, 2005). More extensive results obtained with NMT-Keras can be found at Peris and Casacuberta (2018a). For the first task, we used the standard partitions. For the Europarl corpus, we used the `newstest2012` and `newstest2013`, as development and test, in order to establish a comparison with other IMT works (e.g. Ortiz-Martínez, 2016). The NMT system was configured as in Peris and Casacuberta (2018a). For the sake of comparison, we include results of phrase-based statistical machine translation (PB-SMT), using the standard setup of `Moses` (Koehn et al., 2007). We computed significance tests (95%) via approximate randomization and confidence intervals with bootstrap resampling (Riezler and Maxwell, 2005).

Table 1 shows the translation quality. NMT-Keras outperformed `Moses`, obtaining significant TER and BLEU improvements almost every language pair. Only in one case `Moses` obtained better TER than NMT-Keras.

		TER (↓)		BLEU (↑)	
		PB-SMT	NMT	PB-SMT	NMT
EU	En→De	54.1 ± 1.9	$\mathbf{52.3 \pm 1.9}$	35.4 ± 2.1	$\mathbf{36.4 \pm 2.0}$
	En→Fr	41.4 ± 1.6	$\mathbf{38.4 \pm 1.5}$	47.8 ± 1.7	$\mathbf{50.4 \pm 1.6}$
Europarl	En→De	$\mathbf{62.2 \pm 0.3}$	63.1 ± 0.4	19.5 ± 0.3	$\mathbf{20.0 \pm 0.3}$
	En→Fr	56.1 ± 0.3	$\mathbf{55.0 \pm 0.3}$	26.5 ± 0.3	$\mathbf{27.8 \pm 0.3}$

Table 1. Results of translation quality for all tasks in terms of TER [%] and BLEU [%]. Results that are statistically significantly better for each task and metric are boldfaced. Extended results can be found in Peris and Casacuberta (2018a).

4.1. Interactive NMT

We evaluate now the performance of NMT-Keras on an IMT scenario. We are interested in measuring the effort that the user must spend in order to achieve the desired translation. Due to the prohibitive cost that an experimentation with real users conveys, the users were simulated. The references of our datasets were considered to be the desired translations. The amount of effort was measured according to keystroke mouse-action ratio (KSMR) (Barrachina et al., 2009), defined as the number of keystrokes plus the number of mouse-actions required for obtaining the desired sentence, divided by the number of characters of such sentence.

		KSMR [%] (↓)	
		INMT	SOTA
EU	En→De	**19.8 ± 0.5**	30.5 ± 1.1‡
	En→Fr	**16.3 ± 0.4**	25.5 ± 1.1‡
Europarl	En→De	**32.9 ± 0.2**	49.2 ± 0.4†
	En→Fr	**29.8 ± 0.2**	44.4 ± 0.5†

Table 2. Effort required by INMT systems compared to the state-of-the-art (SOTA), in terms of KSMR [%]. Results that are statistically significantly better for each task and metric are boldfaced. † refers to Ortiz-Martínez (2016), ‡ to Barrachina et al. (2009). Extended results can be found in Peris and Casacuberta (2018a).

Table 2 shows the performance in KSMR (%) of the INMT systems. We also compare these results with the best results obtained in the literature for each task. All INMT systems outperformed by large the other ones. Again, we refer the reader to Peris and Casacuberta (2018a) for a larger set of experiments, including OL.

5. Conclusions

We introduced NMT-Keras, a toolkit built on the top of Keras, that aims to ease the deployment of complex NMT systems by having a modular and extensible design. NMT-Keras has a strong focus on building adaptive and interactive NMT systems; which leverage the effort reduction of a user willing to obtain high-quality translations. Finally, its flexibility allows the NMT-Keras framework to be applied directly to other problems, such as multimedia captioning or sentence classification.

We intend to continue the active development of the tool, including new functionalities and improving the quality of the source code. Moreover, we hope to integrate our tool into the Keras ecosystem in a near future.

Acknowledgements

Much of our Keras fork and the Multimodal Keras Wrapper libraries were developed together with Marc Bolaños. We also acknowledge the rest of contributors to these open-source projects. The research leading this work received funding from grants PROMETEO/2018/004 and CoMUN-HaT - TIN2015-70924-C2-1-R. We finally acknowledge NVIDIA Corporation for the donation of GPUs used in this work.

Bibliography

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of USENIX-OSDI*, volume 16, pages 265–283, 2016. URL https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473*, 2015. URL http://arxiv.org/abs/1409.0473.

Barrachina, Sergio, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical Approaches to Computer-Assisted Translation. *Computational Linguistics*, 35(1):3–28, 2009. doi: 10.1162/coli.2008.07-055-R2-06-29. URL https://doi.org/10.1162/coli.2008.07-055-R2-06-29.

Bolaños, Marc and Petia Radeva. Simultaneous food localization and recognition. In *ICPR*, pages 3140–3145, 2016. doi: 10.1109/ICPR.2016.7900117.

Bolaños, Marc, Aina Ferrà, and Petia Radeva. Food Ingredients Recognition Through Multi-label Learning. In *ICIAP*, pages 394–402, 2017. doi: 10.1007/978-3-319-70742-6_37.

Bolaños, Marc, Álvaro Peris, Francisco Casacuberta, and Petia Radeva. VIBIKNet: Visual bidirectional kernelized network for visual question answering. In *Proceedings of IbPRIA*, pages 372–380, 2017. doi: 10.1007/978-3-319-58838-4_41.

Bolaños, Marc, Álvaro Peris, Francisco Casacuberta, Sergi Soler, and Petia Radeva. Egocentric video description based on temporally-linked sequences. *Journal of Visual Communication and Image Representation*, 50:205–216, 2018. doi: 10.1016/j.jvcir.2017.11.022. URL https://doi.org/10.1016/j.jvcir.2017.11.022.

Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv:1504.00325*, 2015. URL http://arxiv.org/abs/1504.00325.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the Workshop on SSST*, pages 103–111, 2014. URL http://www.aclweb.org/anthology/W14-4012.

Chollet, François et al. Keras. https://github.com/keras-team/keras, 2015. GitHub repository.

Duchi, John, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. URL http://dl.acm.org/citation.cfm?id=2021068.

Gelbart, Michael A., Jasper Snoek, and Ryan P. Adams. Bayesian Optimization with Unknown Constraints. In *Proceedings of UAI*, pages 250–259. AUAI Press, 2914. ISBN 978-0-9749039-1-0. URL http://dl.acm.org/citation.cfm?id=3020751.3020778.

Glorot, Xavier and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256, 2010. URL http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of ICCV*, pages 1026–1034, 2015.

Hochreiter, Sepp and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Ioffe, Sergey and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. URL http://arxiv.org/abs/1502.03167.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of ACL*, pages 1–10, 2015. doi: 10.3115/v1/P15-1001. URL http://www.aclweb.org/anthology/P15-1001.

Kingma, Diederik and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. URL http://arxiv.org/abs/1412.6980.

Klambauer, Günter, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, pages 971–980, 2017. URL http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017. URL http://www.aclweb.org/anthology/P17-4012.

Knowles, Rebecca and Philipp Koehn. Neural Interactive Translation Prediction. In *Proceedings of the AMTA*, pages 107–120, 2016. URL https://www.cs.jhu.edu/~phi/publications/neural-interactive-translation.pdf.

Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit*, pages 79–86, 2005. URL http://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL*, pages 177–180, 2007. URL http://dl.acm.org/citation.cfm?id=1557769.1557821.

Luong, Thang, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-Based Neural Machine Translation. In *Proceedings of EMNLP*, pages 1412–1421, 2015. doi: 10.18653/v1/D15-1166. URL http://www.aclweb.org/anthology/D15-1166.

Ortiz-Martínez, Daniel. Online Learning for Statistical Machine Translation. *Computational Linguistics*, 42(1):121–161, 2016. doi: 10.1162/COLI_a_00244.

Pascanu, Razvan, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to Construct Deep Recurrent Neural Networks. *arXiv:1312.6026*, 2014. URL https://arxiv.org/abs/1312.6026.

Peris, Álvaro and Francisco Casacuberta. Online Learning for Effort Reduction in Interactive Neural Machine Translation. *arXiv:1802.03594*, 2018a. URL http://arxiv.org/abs/1802.03594.

Peris, Álvaro and Francisco Casacuberta. Active Learning for Interactive Neural Machine Translation of Data Streams. *arXiv:1807.11243*, 2018b. URL https://arxiv.org/abs/1807.11243.

Peris, Álvaro, Marc Bolaños, Petia Radeva, and Francisco Casacuberta. Video Description using Bidirectional Recurrent Neural Networks. In *Proceedings of ICANN*, pages 3–11, 2016. doi: 10.1007/978-3-319-44781-0. URL https://doi.org/10.1007/978-3-319-44781-0_1.

Peris, Álvaro, Luis Cebrián, and Francisco Casacuberta. Online Learning for Neural Machine Translation Post-editing. *arXiv:1706.03196*, 2017a. URL http://arxiv.org/abs/1706.03196.

Peris, Álvaro, Mara Chinea-Ríos, and Francisco Casacuberta. Neural Networks Classifier for Data Selection in Statistical Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, 1(108):283–294, 2017b. doi: 10.1515/pralin-2017-0027.

Peris, Álvaro, Miguel Domingo, and Francisco Casacuberta. Interactive neural machine translation. *Computer Speech & Language*, 45:201–220, 2017c. URL https://doi.org/10.1016/j.csl.2016.12.003.

Riezler, Stefan and John T Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the workshop on MTSE*, pages 57–64, 2005. URL http://www.aclweb.org/anthology/W05-0908.

Schuster, Mike and Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the ACL*, pages 1715–1725, 2016. doi: 10.18653/v1/P16-1162. URL http://www.aclweb.org/anthology/P16-1162.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations at EACL*, pages 65–68, 2017. URL http://aclweb.org/anthology/E17-3017.

Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*, volume 27, pages 3104–3112, 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of CVPR*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688*, 2016. URL https://arxiv.org/abs/1605.02688.

Tieleman, Tijmen and Geoffrey Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Proceedings of NIPS*, volume 30, pages 5998–6008, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vaswani, Ashish, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. Tensor2tensor for neural machine translation. *arXiv:1803.07416*, 2018. URL https://arxiv.org/abs/1803.07416.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICLR*, pages 2048–2057, 2015. URL http://proceedings.mlr.press/v37/xuc15.html.

Zeiler, Matthew D. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701*, 2012. URL https://arxiv.org/abs/1212.5701.

Zhou, Bolei, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of CVPR*, pages 2921–2929, 2016. doi: 10.1109/CVPR.2016.319.

Address for correspondence:
Álvaro Peris
lvapeab@prhlt.upv.es
Pattern Recognition and Human Language Technology Research Center,
Universitat Politècnica de València,
Camino de Vera s/n, 46022 Valencia, SPAIN.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018 125–135

Open Source Toolkit for Speech to Text Translation

Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller,
Ngoc-Quan Pham, Sebastian Stüker, Alex Waibel

Karlsruhe Institute of Technology, Faculty of Informatics, Interactive Systems Labs, Karlsruhe, Germany

Abstract

In this paper we introduce an open source toolkit for speech translation. While there already exists a wide variety of open source tools for the essential tasks of a speech translation system, our goal is to provide an easy to use recipe for the complete pipeline of translating speech. We provide a Docker container with a ready to use pipeline of the following components: a neural speech recognition system, a sentence segmentation system and an attention-based translation system. We provide recipes for training and evaluating models for the task of translating English lectures and TED talks to German. Additionally, we provide pre-trained models for this task. With this toolkit we hope to facilitate the development of speech translation systems and to encourage researchers to improve the overall performance of speech translation systems.

1. Introduction

In recent years a great part of the research interest for automatic speech recognition (ASR) and machine translation (MT) systems focuses on neural approaches. ASR systems used to rely on separate components like a language model, a pronunciation dictionary and an acoustic model and machine translation systems on phrase based approaches. With the emergence of neural networks the implementation of ASR and MT systems became relatively easy, which lead to a number of open-source toolkits for both tasks. Examples for neural ASR toolkits include Eesen (Miao et al., 2015) and Espnet,[1] while XNMT (Neubig et al., 2018) and openNMT-py (Klein et al., 2017) mainly deal with MT systems.

[1] https://github.com/espnet/espnet

 Corresponding author: jan.niehues@kit.edu
Cite as: Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller, Ngoc-Quan Pham, Sebastian Stüker, Alex Waibel. Open Source Toolkit for Speech to Text Translation. The Prague Bulletin of Mathematical Linguistics No. 111, 2018, pp. 125–135. doi: 10.2478/pralin-2018-0011.

Most of these toolkits focus on the evaluation of classical ASR or MT tasks. However, when performing speech translation, the intersection between the different components is an important problem, which rarely receives much attention. ASR systems usually output noisy transcripts without casing and punctuation, which makes the task of the MT system more difficult. End-to-end speech translation systems, that directly translate speech, are mostly difficult to train due to the limited amount of parallel speech translation data.

In this paper we present an open-source toolkit[2] that combines the following components:

- A CTC and an attention based ASR system
- A system to generate the punctuation and the casing of the ASR output
- A neural MT system

We provide recipes to both train and evaluate these models for the task of translating English talks to German. We exclusively use open source data for both training and testing. Pre-trained models of each component are available to download, which makes it possible to replace the individual components without much effort and evaluate the performance directly on speech translation tasks.

2. System Description

The speech translation system presented in this work uses a pipeline approach. According to this pipeline, the audio signal is processed by a sequence of different components to generate the translation. In this work, we use a pipeline of three components. First, the audio is processed by an ASR system to generate a transcript in the source language. In the framework, we integrated two different methods to generate the transcripts. They can be generated by a CTC-based system or by an attentional encoder-decoder based system.

In a second step, punctuation and case information are added to the transcript. This is done by a monolingual translation system based on an attentional encoder-decoder model commonly used in neural machine translation.

The third and last step translates the source text into the target language using a neural machine translation system. We describe the details of all the components in detail in this section.

2.1. Speech Recognition

We include two different speech recognition systems, a CTC based system (Graves et al., 2006) and an attention based system (Bahdanau et al., 2015; Chan et al., 2016). For both approaches we use XNMT (Neubig et al., 2018) to extract 40-dimensional log-Mel-filterbank features with per-speaker mean- and variance-normalization. Both

[2] https://github.com/isl-mt/SLT.KIT

systems transcribe utterances. Our training data is already split into utterances. We rely on the LIUM speaker diarization tool (Rouvier et al., 2013) during testing to create the utterances.

2.1.1. Attentional ASR

Our attentional ASR model follows the listen-attend-spell (Chan et al., 2016) architecture and is similar to the system described by Neubig et al. (2018). The model is implemented with XNMT. Our toolkit exclusively uses XNMT for speech recognition tasks. Compared to a conventional neural machine translation architecture, we replace the encoder with a 4-layer bidirectional pyramidal encoder with a total downsampling factor of 8. The layer size is set to 512, the target embedding size is 64, and the attention uses an MLP of size 128. Input to the model are Mel-filterbank features with 40 coefficients. For regularization, we apply variational dropout of rate 0.3 in all LSTMs, and word dropout of rate 0.1 on the target side (Gal and Ghahramani, 2016). We also fix the target embedding norm to 1 (Nguyen and Chiang, 2018). For training, we use Adam (Kingma and Ba, 2015) with initial learning rate of 0.0003, which is decayed by factor 0.5 if no improved WER is observed. To further facilitate training, label smoothing (Szegedy et al., 2016) is applied. For the search, we use beam size 20 and length normalization with the exponent set to 1.5.

2.1.2. CTC-based ASR

Our CTC-based ASR model is similar to the system described by Zenkel et al. (2018). The input to the model are 40-dimensional Mel-filterbank features. We use every third speech feature of our input sequence and choose the start offset during training randomly, which has the advantage of a lower input sequence length. We train the model to predict Byte Pair Units (Sennrich et al., 2016) [BPE].

The CTC based model consists of four bidirectional LSTM layers with 400 units in each direction followed by a softmax layer. The size of the softmax layer depends on the number of different BPE units we create. We use a dropout rate of 0.25 for all LSTM layers. We train two models based on BPE units with 300 (small model) and 10000 (big model) merges, respectively.

We use SGD with a learning rate of 0.0005 and a momentum term of 0.9 for training. The learning rate is halved whenever the validation token error rate does not decrease by more than 0.1%. We first train the small model and initialize the parameters of the LSTM layers of the big model with the smaller model. We decode the model by greedily selecting the most likely output at each frame.

2.2. Punctuation

Automatic speech recognition (ASR) systems typically do not generate punctuation marks or reliable casing. Using the raw output of these systems as input to MT

causes a performance drop due to mismatched train and test conditions. To create segments and better match typical MT training conditions, we use a monolingual NMT system to add sentence boundaries, insert proper punctuation, and add case where appropriate before translating (Cho et al., 2017).

For both, the punctuation system and the machine translation system, we use the openNMT-py toolkit (Klein et al., 2017) to train the models and to generate the translation. The main difference between the punctuation system and the machine translation system is the input and output data used for training and testing.

The idea of the monolingual machine translation system is to translate from lowercased, unpunctuated text into text with case information and punctuation. Since we do not have any information about the sentence boundaries when inserting the punctuation and case information, we also remove them from the training data. Therefore, in the first step of the pre-processing, we randomly segment the source corpus of the training data into chunks of 20 to 30 words. Based on this randomly segmented corpus, we build the input and output data for the monolingual translation system.

For the input data, we remove all punctuation marks and lowercase all words. Since we will get lower-cased input, we cannot use the same byte-pair encoding (Sennrich et al., 2016) as for the machine translation system. Therefore, we train a separate byte-pair encoding on the lower-cased source data with a code size of 40k. To summarize, the source sequence consists of lower-cased BPE units without any punctuation.

For the target side, we do not want to change the words in the output sentence, but only add case and punctuation information. Therefore, we replace the sentence by features indicating case with punctuation attached. Every word is replaced by a letter *U* or *L*, whether it is upper-cased or lower-cased. Furthermore, punctuation marks following the word are directly attached to the letter.

For example, if the training segment is *I felt worse. Why? I wrote a whole book*, the source input sequence could be *i felt wor@@ se why i wro@@ te a who@@ le book* and the target output sequence will be *U L L. U? U L L L L*.

Based on this method, a translation system is trained to transform the input text into the output tokens. By default, we use the same setup as for the translation system between source and target language.

At test time, we follow the sliding window technique described by Cho et al. (2012). Therefore, we created a test set with segments of length 10 starting with every word on the input data. This means, that except for the beginning and the end of the document, every word occurs ten times, at all positions within the segment. This of course dramatically increases the number of sentences in the test data. In a second step, we generate the target features by applying the monolingual translation system. In a post-processing step, we case the word as it most frequently occurs in the output. We insert punctuation marks, if there is at least one punctuation mark after the word in one of the 10 segments containing this word. If different punctuation marks are predicted, we take the most frequent one. Finally, if the punctuation mark is an end of sentence punctuation mark *{".","!","?"}*, we also start a new segment. The seg-

mented test data with case and punctuation information is passed on to the machine translation system.

2.3. MT

For machine translation, we use a neural machine translation system trained with openNMT-py. By default, we use a rnn-based system.

Within the toolkit, we provide recipes to train a rather small sized translation system. The translation system is trained only on the TED corpus. Due to the limited training data size, we use a smaller model and set the hidden size of the word embeddings as well as for the LSTMs to 512. Furthermore, we use dropout in all the models with a dropout rate of 0.2. In the first training step, we train the model for 10 epochs using Adam optimization. We perform early stopping by evaluating the model after each epoch on the validation data. In the second step, we continue to train the system using a lower learning rate of 0.000125 for another 5 epochs.

The training scripts for the NMT models are provided in *SLT.KIT/scripts/openNMT-py*.

3. Training Data

3.1. ASR

For training the speech recognition systems we use the version 2 of the Tedlium Corpus (Rousseau et al., 2014). We store the log-Mel-filterbank features of each utterances in a hdf5 file.[3] The transcription is stored in a text file one utterance at a time. The key to access the features in the hdf5 file matches the line number of the transcription in the text file. We do not use any additional language modeling data.

3.2. Punctuation and Machine Translation

The machine translation system and the punctuation system are trained on parallel data. We use the TED corpus (Cettolo et al., 2012)[4] and the proceedings from the European Parliament (Koehn, 2005).[5] The source and target sentences are stored in individual text files. As validation data for all systems, we use the dev2010 set provided by IWSLT evaluation campaign (Cettolo et al., 2014).

Prior to translation, we pre-process the data. The default preprocessing includes tokenization, true-casing and byte-pair encoding. The tokenization and true-casing uses the tools from the moses toolkit. The byte-pair encoding is trained on all parallel

[3] https://github.com/h5py/h5py

[4] https://wit3.fbk.eu/

[5] http://statmt.org/europarl/

data and joint codes for both languages are learned. By default, we use a code size of 40k. The scripts to train the true-casing model, to learn the byte-pair encoding and to apply the models to the test data can be found in *SLT.KIT/scripts/defaultPreprocessor*.

4. Evaluation

4.1. Dataset

We evaluate the performance of the ASR and MT systems on English TED talks and its translation to German. We use the test sets used for the IWSLT conference (Cettolo et al., 2014), which are publicly available.[6] We test the input on the following test sets: dev2010, tst2010, tst2013, tst2014.

4.2. Evaluation Metrics

We use Sclite[7] for scoring the ASR output. We calculate the Word Error Rate (WER) based on the provided references in the test sets. Because we do not get a segmentation into utterances, we use talks as the segments for scoring.

In contrast to text translation, the segmentation into sentences is not given a priori in speech translation tasks. Therefore, the standard MT evaluation metrics cannot be applied directly. In this framework, we use the mwerSegmenter[8] to segment the output of the speech translation system according to the reference. In a second step, we can then calculate all machine translation evaluation metrics on the re-segmented output of the translation system.

To evaluate the output, we calculate four different metrics. We generate the BLEU score (Papineni et al., 2002), the TER score (Snover et al., 2006), the BEER metric (Stanojevic and Sima'an, 2014) and CharacTER (Wang et al., 2016). While these metrics are all calculated considering case-information, we also calculate case-insensitive (ci) BLEU and TER scores.

In Table 1 you can find an example sentence processed by our toolkit. First of all the speaker diarization tool generates utterances. These utterances are transcribed by the ASR system. The utterances do not consist of single sentences. The segmentation tool re-segments the output and adds punctuation. In the example this leads to a slightly longer sentence than in the reference, because the expressions "I think" and "we know" are segmented differently. This sentence is then translated to German by the MT system.

[6] https://sites.google.com/site/iwsltevaluation2018/Lectures-task

[7] http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

[8] https://www-i6.informatik.rwth-aachen.de/web/Software/mwerSegmenter.tar.gz

Step	Text
Reference EN	We know that the first 10 years of a career has an exponential impact on how much money you're going to earn.
ASR EN	… don't panic I think this crowd is going to be thought | i think we know that the first ten years of a career has an | exponential impact on how much money you're going turn we know that more than half of | Americans are married …
Punctuation	I think we know that the first ten years of a career has an exponential impact on how much money you're going turn we know.
MT DE	ich denke, wir wissen, dass die ersten 10 Jahre einer Karriere einen entscheidenden Einfluss darauf haben, wie viel Geld wir wissen.
Reference DE	Wir wissen, dass die ersten 10 Jahre eines Berufes eine exponentielle Auswirkung darauf haben, wie viel Geld man verdienen wird.

Table 1. References of an example source and target sentence and the output of the different steps of our system. Utterance boundaries of the ASR system are visualized with the token "|".

4.3. Results

We evaluate the performance of the ASR systems before performing any translation. In Table 2 we report results for the attention based system [Attention], the CTC system with 300 BPE merges [CTC 300] and the CTC system with 10k merges [CTC 10k]. We additionally combine the outputs of the three system by using Rover (Fiscus) [Rover].

Since we do not throw away any segments of the audio file, it still contains segments containing silence. The attention based system tries to produce output for these segments, which leads to a high number of insertion errors. On the other hand, the CTC model handles this situation well and outputs an empty transcript. The CTC 300 model has a higher error rate, which is mostly due to misspelling of words. This can be fixed by training an additional language model as in Zenkel et al. (2018). Combining all three systems improves the results and yields balanced insertion and deletion errors. Many errors of the ASR system are also due to normalization issues between the reference and the hypothesis, especially numbers and dates cause many errors.

Category	S	D	I	WER
Attention	17.5%	2.9%	8.8%	29.2%
CTC 300	17.3%	5.8%	3.6%	26.7%
CTC 10k	13.5%	6.0%	3.3%	22.8%
Rover	13.1%	3.9%	4.2%	21.2%

Table 2. Substitution (S), Insertion (I), Deletion (D) and Word Error Rate (WER) on test2014 for different ASR systems

Category	BLEU	TER	BEER	CharacTER	BLEU(ci)	TER(ci)
Attention	11.88	79.75	41.49	76.98	12.57	77.90
CTC 300	11.49	76.79	40.57	81.52	12.18	75.12
CTC 10k	12.58	74.16	42.05	81.90	13.34	72.36
Rover	13.28	74.34	42.43	78.38	14.01	72.62

Table 3. MT scores on tst2014 for various ASR outputs

We present the results of the MT system in Table 3. Better ASR results also lead to better MT results for all presented metrics. The only exception is the attention based system, which gets better scores than the CTC 300 output when evaluating the performance of the resulting MT output.

Category	dev2010	tst2010	tst2013	tst2014
Attention	13.42	13.57	12.04	11.88
CTC 300	12.33	11.88	12.47	11.49
CTC 10k	13.04	13.44	13.41	12.58
Rover	13.98	14.08	13.73	13.28

Table 4. BLEU scores for different ASR outputs on all test sets

In Table 4 we state the results on the remaining test sets. We do not see a consistent trend if the attention based ASR system or the CTC based system performs better. However, the bigger CTC model consistently yields better BLEU scores than the small CTC model. This can be explained by the higher word error rate and a significantly higher number of miss-spellings in the output of the small model. While the attention based model also yields higher word error rates than the CTC 10k model, this is mostly

due to the higher number of insertion and does not hurt the translation performance as much. We additionally notice that combining the outputs of all ASR systems with Rover improves the results across all test sets.

5. Conclusion

This paper has introduced a toolkit for speech translation. It consistently uses open-source software as well as freely available training and test data. We presented results on test sets for pre-trained models for English to German speech translation. By additionally open-sourcing the trained models we hope to facilitate the task of improving individual components for speech translation systems.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015. URL https://arxiv.org/pdf/1409.0473v7.pdf.

Cettolo, Mauro, Christian Girardi, and Marcello Federico. WIT^3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the 16$^{\text{th}}$ Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy, 2012. URL http://hltshare.fbk.eu/EAMT2012/html/Papers/59.pdf.

Cettolo, Mauro, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *Proceedings of the 11th International Workshop on Spoken Language Translation*, Lake Tahoe, California, 2014. URL http://www.mt-archive.info/10/IWSLT-2014-Cettolo.pdf.

Chan, William, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2016. doi: 10.1109/icassp.2016.7472621. URL https://doi.org/10.1109%2Ficassp.2016.7472621.

Cho, E., J. Niehues, and A. Waibel. Segmentation and punctuation prediction in speech language translation using a monolingual translation system. In *Proceedings of the Ninth International Workshop on Spoken Language Translation (IWSLT)*, 2012. URL https://isl.anthropomatik.kit.edu/pdf/Cho2012.pdf.

Cho, Eunah, Jan Niehues, and Alex Waibel. NMT-Based Segmentation and Punctuation Insertion for Real-Time Spoken Language Translation. In *Interspeech 2017*. ISCA, aug 2017. doi: 10.21437/interspeech.2017-1320. URL https://doi.org/10.21437%2Finterspeech.2017-1320.

Fiscus, J.G. A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE. doi: 10.1109/asru.1997.659110. URL https://doi.org/10.1109%2Fasru.1997.659110.

Gal, Yarin and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Neural Information Processing Systems*

Conference (NIPS), Barcelona, Spain, 2016. URL https://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networks.pdf.

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, 2006. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145%2F1143844.1143891.

Kingma, Diederik P. and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, Banff, Canada, 2015. URL https://arxiv.org/pdf/1412.6980.pdf.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*. Association for Computational Linguistics, 2017. doi: 10.18653/v1/p17-4012. URL https://doi.org/10.18653%2Fv1%2Fp17-4012.

Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, 2005. URL https://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf.

Miao, Yajie, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, dec 2015. doi: 10.1109/asru.2015.7404790. URL https://doi.org/10.1109%2Fasru.2015.7404790.

Neubig, Graham, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, et al. XNMT: The extensible neural machine translation toolkit. *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase.*, 2018. URL https://arxiv.org/pdf/1803.00188v1.pdf.

Nguyen, Toan and David Chiang. Improving Lexical Choice in Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1031. URL https://doi.org/10.18653%2Fv1%2Fn18-1031.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.pdf.

Rousseau, Anthony, Paul Deléglise, and Yannick Esteve. Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. In *LREC*, 2014. URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/1104_Paper.pdf.

Rouvier, Mickael, Grégor Dupuy, Paul Gay, Elie Khoury, Teva Merlin, and Sylvain Meignier. An open-source state-of-the-art toolbox for broadcast news diarization. In *Interspeech*, 2013. URL http://publications.idiap.ch/downloads/reports/2013/Rouvier_Idiap-RR-33-2013.pdf.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/p16-1162. URL https://doi.org/10.18653%2Fv1%2Fp16-1162.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, 2006. URL https://www.cs.umd.edu/~snover/pub/amta06/ter_amta.pdf.

Stanojevic, Milos and Khalil Sima'an. BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2014. doi: 10.3115/v1/w14-3354. URL https://doi.org/10.3115%2Fv1%2Fw14-3354.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016. doi: 10.1109/cvpr.2016.308. URL https://doi.org/10.1109%2Fcvpr.2016.308.

Wang, Weiyue, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. CharacTer: Translation Edit Rate on Character Level. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/w16-2342. URL https://doi.org/10.18653%2Fv1%2Fw16-2342.

Zenkel, Thomas, Ramon Sanabria, Florian Metze, and Alex Waibel. Subword and Crossword Units for CTC Acoustic Models. *Interspeech*, 2018. URL https://arxiv.org/pdf/1712.06855.pdf.

Address for correspondence:
Jan Niehues
jan.niehues@kit.edu
Karlsruhe Institute of Technology
Adenauerring 2,
76131 Karlsruhe, Germany

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 111 OCTOBER 2018

INSTRUCTIONS FOR AUTHORS

Manuscripts are welcome provided that they have not yet been published elsewhere and that they bring some interesting and new insights contributing to the broad field of computational linguistics in any of its aspects, or of linguistic theory. The submitted articles may be:

- long articles with completed, wide-impact research results both theoretical and practical, and/or new formalisms for linguistic analysis and their implementation and application on linguistic data sets, or
- short or long articles that are abstracts or extracts of Master's and PhD thesis, with the most interesting and/or promising results described. Also
- short or long articles looking forward that base their views on proper and deep analysis of the current situation in various subjects within the field are invited, as well as
- short articles about current advanced research of both theoretical and applied nature, with very specific (and perhaps narrow, but well-defined) target goal in all areas of language and speech processing, to give the opportunity to junior researchers to publish as soon as possible;
- short articles that contain contraversing, polemic or otherwise unusual views, supported by some experimental evidence but not necessarily evaluated in the usual sense are also welcome.

The recommended length of long article is 12–30 pages and of short paper is 6–15 pages.

The copyright of papers accepted for publication remains with the author. The editors reserve the right to make editorial revisions but these revisions and changes have to be approved by the author(s). Book reviews and short book notices are also appreciated.

The manuscripts are reviewed by 2 independent reviewers, at least one of them being a member of the international Editorial Board.

Authors receive a printed copy of the relevant issue of the PBML together with the original pdf files.

The guidelines for the technical shape of the contributions are found on the web site `http://ufal.mff.cuni.cz/pbml`. If there are any technical problems, please contact the editorial staff at `pbml@ufal.mff.cuni.cz`.