PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 109 OCTOBER 2017

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 109 OCTOBER 2017

CONTENTS

Articles

The Prague Bulletin of Mathematical Linguistics
NUMBER 109 OCTOBER 2017 5–14

Open-Source Neural Machine Translation API Server

Sander Tars, Kaspar Papli, Dmytro Chasovskyi, Mark Fishel

Institute of Computer Science, University of Tartu, Estonia

Abstract

We introduce an open-source implementation of a machine translation API server. The aim of this software package is to enable anyone to run their own multi-engine translation server with neural machine translation engines, supporting an open API for client applications. Besides the hub with the implementation of the client API and the translation service providers running in the background we also describe an open-source demo web application that uses our software package and implements an online translation tool that supports collecting translation quality comparisons from users.

1. Introduction

The machine translation community boasts numerous open-source implementations of neural (e.g. Junczys-Dowmunt et al., 2016; Sennrich et al., 2017; Helcl and Libovický, 2017; Vaswani et al., 2017), statistical (e.g. Koehn et al., 2007) and rule-based (e.g. Forcada et al., 2011) translation systems. Some of these (e.g. Koehn et al., 2007; Junczys-Dowmunt et al., 2016) even include functionality of server-mode translation, keeping the trained model(s) in memory and responding to the client application's translation requests. However, in most cases the frameworks are tuned for machine translation researchers, and basic production functionality like pre-processing and post-processing pipelines before/after the translation are missing in the translation server implementations.

We present an open-source implementation of a machine translation production server implemented in a modular framework. It supports multiple translation clients running the translation for different language pairs and text domains. The framework consists of:

Corresponding author: fishel@ut.ee
Cite as: Sander Tars, Kaspar Papli, Dmytro Chasovskyi, Mark Fishel. Open-Source Neural Machine Translation API Server. The Prague Bulletin of Mathematical Linguistics No. 109, 2017, pp. 5–14.
doi: 10.1515/pralin-2017-0034.

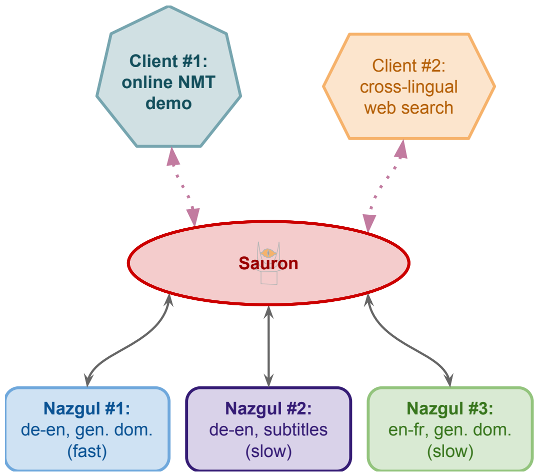


Figure 1. The overall architecture is very simple. Sauron is the server hub, satisfying requests from client applications by querying the translation providers, the Nazgul.

- Sauron: a translation server hub, receiving client requests to translate a text using one of pre-configured translation engines, the Nazgul,
- Nazgul: a translation provider and engine wrapper with custom pre-processing and post-processing steps before/after the translation,
- and a demo web page that uses these two to serve translations to web users, and includes unbiased feedback collection from the users.

The overall architecture is extremely simple and is shown on Figure 1. The hub (Sauron) can serve several clients and is connected to several instances of Nazgul, the translation providers. Each Nazgul is configured to deliver translations for a specific language pair and possibly text domain.

The structure of this paper is the following. Sauron, the translation server hub, is presented in Section 2. Nazgul, the translation engine wrapper is covered in Section 3. The demo web application is described in Section 4. Finally we refer to related work in Section 5 and conclude the paper in Section 6.

2. Sauron, the Translation Server Hub

The central hub tying together all of the components of our framework is Sauron. It works as a reverse proxy, receiving translation requests from client applications and retrieving the translations from one of the Nazgul (which are described in Section 3). The code is freely available on GitHub.[1]

The main features of this central component include

- support for multiple language pairs and text domains
- asynchronous processing of simultaneous translation requests, to enable efficient processing in stressful environments with several requests per second or more
- support for authentication to limit the service only to registered clients if desired
- letting the client application choose between single sentence or whole text translation speed priority

2.1. Client Interface

Access to a running Sauron server is implemented as a simple REST API. Once deployed it runs at a specified URL/IP address and port and supports both GET and POST HTTP communication methods. The API is described and can be tested online on SwaggerHub.[2] The input parameters are:

auth	the authentication token, set in configuration
langpair	a identifier of the source-target language pair, set in configuration
src	the source text
domain	text domain identifier; it can be omitted, leading to the usage of a general-domain translation engine, set in configuration
fast	True indicates the fast, sentence speed-oriented translation method; default is false, document speed-oriented translation
tok	true by default, indicates whether to tokenize the input text
tc	true by default, indicates whether to apply true-casing to the input text
alignweights	false by default, indicates whether to also compute and return the attention weights of the NMT decoder

Although the `fast` parameter is open to interpretation, the idea is to run "fast" translation servers on GPUs, enabling one to focus on the speed of translating a single sentence, while the "slot" servers can be run on CPUs, enabling one to translate a whole document as a batch in multiple threads.

Each combination of language pair, domain and fast/slow has to be covered by a corresponding Nazgul instance, there is no automatic backoff from slow to fast or from in-domain to general domain translation.

[1] https://github.com/TartuNLP/sauron

[2] https://app.swaggerhub.com/apis/kspar/sauron/v1.0

2.2. Configuration

The only configuration required for Sauron is a list of Nazgul translation provider servers. These are described in an XML file located at `$ROOT/src/main/resources/providers.xml`. Each provider is described with the following parameters:

name	The name, used for system identification in logs
languagePair	A string identifier representing the source-target translation language pair; there is no enforced format but the same string must be used as the value for the API request parameter `lang-pair`
translationDomain	A string identifier representing the translation domain; this is similarly mapped to the API request parameter `domain`
fast	The GPU/CPU preference, a boolean indicating whether the server is using a GPU for translation (whether it is fast); this is mapped to the API request parameter `fast`
ipAddress	The IP address of the translation server
port	The listening port of the translation server

2.3. Deployment

Sauron runs on Java Spring Boot.[3] The preferred method of deployment is to use Gradle[4] to build a `war` file:

```
./gradlew war
```

and deploy it into a Java web container such as Tomcat. You can also run the server without a web container:

```
./gradlew bootRun
```

3. Nazgul, the Translation Servant

Nazgul implements a translation server provider for Sauron. Its design is a modular architecture: every step of the translation service process like pre-processing, translating, post-processing, can be easily modified and substituted. The modularity and open-source format is important for usable machine translation to reduce the

[3]`https://projects.spring.io/spring-boot/`

[4]`https://gradle.org/`

time required to create various application specific services. The code for Nazgul is freely available on GitHub.[5]

Nazgul uses AmuNMT/Marian (Junczys-Dowmunt et al., 2016) as the translation engine (though the modularity of the architecture allows one to replace it easily). The main motivation behind it is because it offers fast neural translation. Moreover, we use a particular modification of this software (available on GitHub[6]), which supports extracting the attention weights after decoding.

3.1. Dependencies

Nazgul is written in Python 2.7 for the reasons of broader compatibility. The implementation requires the following dependencies to be satisfied:

- Downloaded and compiled clone of Marian(AmuNMT) with attention weight output
- The NLTK Python library (Bird et al., 2009). More precisely, the modules `punkt`, `perluniprops` and `nonbreaking_prefixes` are needed. NLTK is used for sentence splitting, tokenization and detokenization[7]

The instructions on how to satisfy these dependencies can be found on the Nazgul GitHub page.[8]

3.2. Deployment

With the dependency requirements satisfied, the server can be run from the command-line simply as a Python file. Example command:

```
python nazgul.py -c config.yml -e truecase.mdl -s 12345
```

The command-line options for running are:

-c	configuration file to be used for AmuNMT run
-e	name of the truecasing model file
-s	the port on which the server will listen (default: 12345)

[5]https://github.com/TartuNLP/nazgul

[6]https://github.com/barvins/amunmt

[7]To be precise, NLTK uses Moses (Koehn et al., 2007) to tokenize and detokenize by having a Python module nltk.tokenize.moses wrap the Moses tokenizing scripts.

[8]https://github.com/TartuNLP/nazgul

The true-caser expects the true-casing models to be trained using the Moses true-caser script.[9] The true-casing model file is expected to be in the same directory with the Nazgul.

The configuration file that is required for AmuNMT translation, is also expected to be in the same directory with the Nazgul. The configuration file specifies the translation model file, vocabularies, whether to use byte pair encoding (BPE, Sennrich et al., 2015), whether to display attention info and many more options. One possible configuration file that we use, is presented on the Nazgul GitHub page with explanations. Additional information can be found on both the original AmuNMT and cloned GitHub pages.

Currently the BPE is only available in Nazgul through AmuNMT configuration file. The reason is that in our experiments having BPE through AmuNMT resulted in faster translation. We are also adding support for separate BPE. To train and apply BPE we used the open-source implementation by Sennrich et al. (2015).[10]

3.3. Workflow

This section describes what happens when Nazgul is started and used to translate. The process is implemented in the file `nazgul.py`.

First, it initialises the key components: AmuNMT, tokenizer, detokenizer, true-caser and finally binds a socket to the specified port to listen for translation requests. Nazgul is capable of serving multiple clients simultaneously.

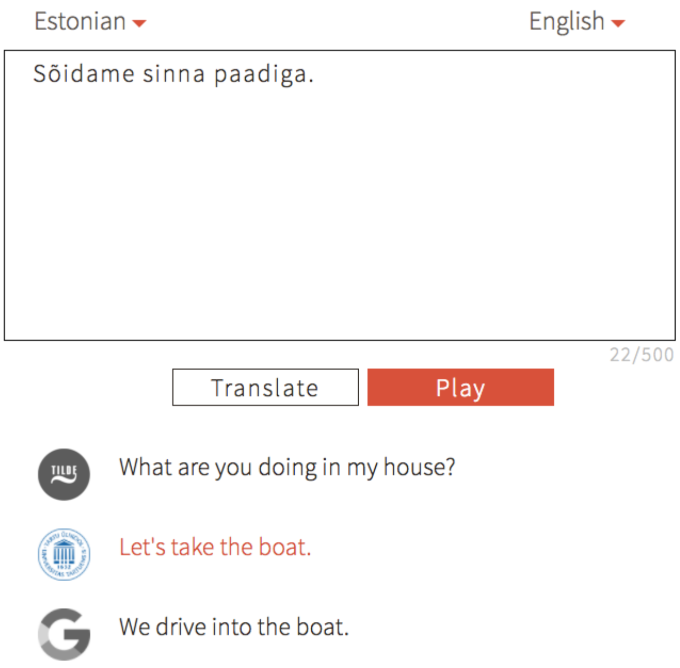
Secondly, when a client connects to Nazgul, the connection is verified and then translation requests are accepted. The necessary protocols are implemented in Sauron, so it is the most convenient option for connecting with Nazgul. For each client connection Nazgul creates a separate thread. The translation request format is a dict in JSON, which includes the fields **src**, **tok** and **tc** that are passed unchanged from Sauron as well as a boolean parameter **alignweights**, which specifies whether this Nazgul should include attention info in the response.

Once the translation request JSON is received, the source string is subjected to pre-processing. Pre-processing starts with sentence splitting, which is always done for the sake of multi-sentence inputs. After that each received sentence is tokenized and truecased, if specified in the JSON input.

After pre-processing, the sentences are sent to the instance of AmuNMT to be translated. From its translation output Nazgul separates the raw translation, attention info, and raw input. It is recommended to disable AmuNMT de-BPE function in the configuration file, otherwise the raw translation will actually be the de-BPEd translation while raw input will be BPEd, thus perturbing the attention info interpretation.

[9] `http://www.statmt.org/moses/?n=Moses.SupportTools#ntoc11`

[10] `https://github.com/rsennrich/subword-nmt`



Sõidame sinna paadiga.

What are you doing in my house?

Let's take the boat.

We drive into the boat.

Figure 2. A screenshot from the web application's Play *functionality, which aims to let the users compare the outputs of three translation engines and also to collect the unbiased feedback from the users' selection of the best translation. The Estonian input reads:* Let's take the boat there.

When the translation output is received, the translated sentences are subjected to post-processing, which includes detokenization (if tokenization is enabled), and de-truecasing.

Finally, the result of the translation process is sent to the client as a utf-8 encoded JSON dict, which includes fields **raw_trans**, **raw_input**, **weights**, and **final_trans**, which is an array of post-processed and de-BPEd translation outputs. The order of the outputs is the same as in the input text after sentence-splitting.

After sending the response JSON, Nazgul waits for either the next request or termination. Anything that is not JSON is interpreted as a termination signal. In Sauron the process is resolved in such a way that after each fulfilled request the connection is closed. The waiting for next requests is a feature for use cases where the bi-directional communication is expected to have a continuous load for several messages, which would make closing and re-opening the connection an unnecessary overhead.

For further reference on communication, refer to both Nazgul and Sauron documentation pages and simple test scripts presented in the GitHub repository.

4. Neurotõlge, the Example Web Application

Finally we describe an NMT web demo implementation that uses Sauron and Nazgul to fulfill translation requests: Neurotõlge.[11] The demo is live at `http://www.neurotolge.ee` (with an international mirror domain `http://neuralmt.ee`), and the code of the implementation is freely available on GitHub.[12]

The basic functionality of the web application is to translate the input text that the client enters. The text can consist of several sentences, and the client can switch between the available source and target languages (English and Estonian in the live version). Once the client presses the "translate" button the text is translated.

4.1. Collecting User Feedback

Beside the "translate" button there is also a "play" button: once pressed, the application uses three different translation engines to translate the source text. In the live version these are the University of Tartu's translator running on Sauron, Google Translate[13] and Tilde Neural Machine Translation.[14]

Once ready all three translations are displayed in random order without telling the user, which output belongs to which translation engine; the user is invited to select the best translation in order to find out which is which. See an example screenshot of this functionality on Figure 2.

The aim of this feedback collection is to get an unbiased estimation of which translation engine gets selected as best most often. Naturally some users will click on the first or on a random translation, but since the order of the translations is random and the identity of the translation engines is hidden, this will only add uniform noise to the distribution of the best translation engines. This approach was inspired by BlindSearch.[15]

4.2. Dependencies

The front-end of the web application is implemented in JavaScript, using AJAX for asynchronous communications with the back-end and the Bootstrap framework[16] for

[11]*Neural machine translation* in Estonian

[12]`https://github.com/TartuNLP/neurotolge`

[13]`http://translate.google.com/`

[14]`https://translate.tilde.com/neural/`

[15]`http://blindsearch.fejus.com/`

[16]`http://getbootstrap.com/`

an appealing graphic design The back-end is built using Flask.[17] It can be connected to any web server, like Apache, or to be run as a standalone server.

5. Related Work

Some MT service frameworks have been introduced for SMT (Sánchez-Cartagena and Pérez-Ortiz, 2010; Federmann and Eisele, 2010; Tamchyna et al., 2013) and designed to work with Moses (Koehn et al., 2007). The Apertium system also includes a web demo and server framework (Forcada et al., 2011).

NeuralMonkey (Helcl and Libovický, 2017) includes server-running mode, and supports several language pairs and text domains (via different system IDs). However, AmuNMT that our framework uses has been shown to run faster and bringing slightly higher translation quality.

6. Conclusions

We introduce an open-source implementation of a neural machine translation API server. The server consists of a reverse proxy or translation hub that accepts translation requests from client applications and an implementation of a back-end translation server with the pre-processing and post-processing pipelines. The current version uses Marian (AmuNMT) as the translation engine, and the modular architecture of the implementation allows it to be replaced with other NMT engines.

We also described a demo web application that uses the API implementation. In addition to letting its users translate texts it also includes a feedback collection component, which can be used to get an idea of the user feedback on the translation quality.

Future work includes adding a database support to the hub implementation to allow the developer to track the usage of the API, as well as a possibility to visualize the alignment matrix of the NMT decoder on the demo web application to help the users analyze translations and understand, why some translations are counter-intuitive.

Acknowledgements

The projects described here were partially supported by the National Programme for Estonian Language Technology, project *EKT88: KaMa: Kasutatav Eesti Masintõlge / Usable Estonian Machine Translation*.[18]

Bibliography

Bird, Steven, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[17] `http://flask.pocoo.org/`

[18] `https://www.keeletehnoloogia.ee/et/ekt-projektid/kama-kasutatav-eesti-masintolge`

Federmann, Christian and Andreas Eisele. MT Server Land: An Open-Source MT Architecure. *The Prague Bulletin of Mathematical Linguistics*, 94:57–66, 2010.

Forcada, Mikel L, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144, 2011.

Helcl, Jindřich and Jindřich Libovický. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17, 2017.

Junczys-Dowmunt, Marcin, Tomasz Dwojak, and Hieu Hoang. Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions. *CoRR*, abs/1610.01108, 2016. URL `http://arxiv.org/abs/1610.01108`.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, 2007.

Sánchez-Cartagena, Víctor and Juan Pérez-Ortiz. ScaleMT: a free/open-source framework for building scalable machine translation web services. *The Prague Bulletin of Mathematical Linguistics*, 93:97–106, 2010.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *CoRR*, abs/1508.07909, 2015. URL `http://arxiv.org/abs/1508.07909`.

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. *CoRR*, abs/1703.04357, 2017. URL `http://arxiv.org/abs/1703.04357`.

Tamchyna, Aleš, Ondřej Dušek, Rudolf Rosa, and Pavel Pecina. MTMonkey: A Scalable Infrastructure for a Machine Translation Web Service. *The Prague Bulletin of Mathematical Linguistics*, 100:31–40, 2013.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Address for correspondence:
Mark Fishel
`fishel@ut.ee`
Institute of Computer Science, University of Tartu
Liivi 2, Tartu 50409
Estonia

The Prague Bulletin of Mathematical Linguistics
NUMBER 109 OCTOBER 2017 15–28

NMTPY: A Flexible Toolkit for Advanced Neural Machine Translation Systems

Ozan Caglayan, Mercedes García-Martínez, Adrien Bardet, Walid Aransa, Fethi Bougares, Loïc Barrault

Laboratoire d'Informatique de l'Université du Maine (LIUM)

Abstract

In this paper, we present *nmtpy*, a flexible Python toolkit based on Theano for training Neural Machine Translation and other neural sequence-to-sequence architectures. *nmtpy* decouples the specification of a network from the training and inference utilities to simplify the addition of a new architecture and reduce the amount of boilerplate code to be written. *nmtpy* has been used for LIUM's top-ranked submissions to WMT Multimodal Machine Translation and News Translation tasks in 2016 and 2017.

1. Introduction

nmtpy is a refactored, extended and Python 3 only version of *dl4mt-tutorial*[1], a Theano (Theano Development Team, 2016) implementation of attentive Neural Machine Translation (NMT) (Bahdanau et al., 2014). The development of *nmtpy* project which has been open-sourced[2] under MIT license in March 2017, started in March 2016 as an effort to adapt *dl4mt-tutorial* to multimodal translation models. *nmtpy* has now become a powerful toolkit where adding a new model is as simple as deriving from an abstract base class, implementing a set of its methods and writing a custom data iterator if necessary. The training and inference utilities are as model-agnostic

[1] https://github.com/nyu-dl/dl4mt-tutorial

[2] https://github.com/lium-lst/nmtpy

Corresponding author: ozancag@gmail.com

as possible allowing one to use them for different sequence generation networks such as multimodal NMT and image captioning to name a few.

Other prominent toolkits in the field are OpenNMT (Klein et al., 2017), Neural Monkey (Helcl and Libovický, 2017) and Nematus (Sennrich et al., 2017). While *nmtpy* and Nematus share the same *dl4mt-tutorial* codebase, the flexibility and the rich set of architectures (Section 3) are what differentiate our toolkit from Nematus. Both OpenNMT and Nematus are solely focused on translation by providing feature-rich but monolithic NMT implementations. Neural Monkey which is based on TensorFlow (Abadi et al., 2016), provides a more generic sequence-to-sequence learning framework similar to *nmtpy*.

2. Design

In this section we first give an overview of a typical NMT training session in *nmtpy* and the design of the translation utility *nmt-translate*. We then describe the configuration file format, explain how to define new architectures and finally introduce the basic deep learning elements and techniques provided by *nmtpy*. A more detailed tutorial about training an NMT model is available on Github [3].

2.1. Training

A training experiment (Figure 1) is launched by providing an INI-style experiment configuration file to *nmt-train* (Listing 1). *nmt-train* then automatically selects a free GPU, sets the seed for NumPy and Theano random number generators, constructs an informative filename for log files and model checkpoints and finally instantiates a Python object of type `"model_type"` given through the configuration file. The tasks of data loading, weight initialization and graph construction are all delegated to this model instance.

```
$ nmt-train -c en-de.conf                                      # Launch an experiment
$ nmt-train -c en-de.conf 'model_type:new_nmt'                 # Override model_type
$ nmt-train -c en-de.conf 'rnn_dim:500' 'embedding_dim:300'    # Change dimensions
$ nmt-train -c en-de.conf 'device_id:gpu5'                     # Force specific GPU device
```

Listing 1. Example usages of nmt-train.

During training, *nmt-train* consumes mini-batches of data from the model's iterator and performs forward/backward passes along with the weight updates. Translation performance on a held-out corpus is periodically evaluated in order to early-stop the training process to avoid overfitting. These periodic evaluations are realized by calling *nmt-translate* which performs beam-search, computes metrics and returns them back to *nmt-train*.

[3] https://github.com/lium-lst/wmt17-mmt

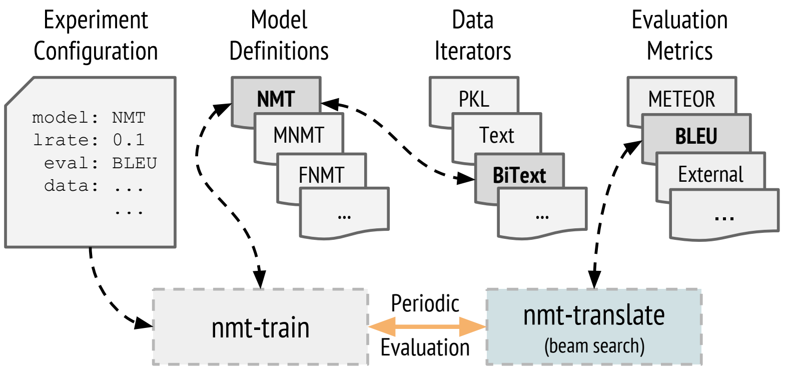


Figure 1. The workflow of a training experiment.

2.2. Translation

nmt-translate performs translation decoding using a beam-search implementation that supports single and ensemble decoding for both monomodal and multimodal translation models (Listing 2).

Since the number of CPUs in a single machine is 2x-4x higher than the number of GPUs and we mainly reserve the GPUs for training, *nmt-translate* makes use of CPU workers for maximum efficiency. More specifically, each worker receives a model instance (or instances when ensembling) and performs the beam-search on samples that it continuously fetches from a shared queue filled by the master process. One thing to note for parallel CPU decoding is that if the installed NumPy is linked against a BLAS implementation with threading support enabled (as in the case with Anaconda & Intel MKL), each spawned process attempts to use all available threads in the machine leading to a resource conflict. In order for *nmt-translate* to benefit correctly from parallelism, the number of threads per process should thus be limited to one [4]. The impact of this setting and the overall decoding speed in terms of words/sec (wps) are reported in Table 1 for a medium-sized En→Tr NMT with ~10M parameters.

```
# Decode on 30 CPUs with beam size 10, compute BLEU/METEOR
$ nmt-translate -j 30 -b 10 -M bleu meteor -m model.npz -S val.bpe.en -R val.de -o out.de
# Generate 50-best list with an ensemble of checkpoints
$ nmt-translate -b 50 -N 50 -m model*npz -S val.tok.de -o out.tok.50best.de
```

Listing 2. Example usages of nmt-translate.

[4]This is achieved by setting `X_NUM_THREADS=1` environment variable where `X` is one of `OPENBLAS,OMP,MKL` depending on the NumPy installation.

# BLAS Threads	Tesla K40	4 CPU	8 CPU	16 CPU
Default	185 wps	25 wps	25 wps	25 wps
Set to 1	185 wps	109 wps	198 wps	332 wps

Table 1. Median beam-search speed over 3 runs with beam size 12: decoding on a single Tesla K40 GPU is roughly equivalent to using 8 CPUs (Intel Xeon E5-2687v3).

2.3. Configuration

Each *nmtpy* experiment is defined with an INI-style configuration file that has four mandatory sections, namely `[training]`, `[model]`, `[model.dicts]` and `[model.data]`. Each section may contain a number of options in `key:value` format where the value can be built-in Python data types like integer, float, boolean, string, list, etc. Paths starting with a tilde are automatically expanded to `$HOME` folder.

The options defined in the `[training]` section are consumed by *nmt-train* while the ones in the `[model.*]` sections are automatically passed to the model instance (specifically, to its `__init__()` method) created by *nmt-train*. This allows one to add a new `key:value` option to the configuration file and access it automatically from the model instance.

Any option defined in the configuration file can be overridden through the command line by passing new `key:value` pair as the last argument to *nmt-train* (Listing 1). The common defaults defined in `nmtpy/defaults.py` are shortly described in Table 2. A complete configuration example is provided in Appendix A.

2.4. Defining New Architectures

A new architecture can be defined by creating a new file (i.e. `my_amazing_nmt.py`) under `nmtpy/models`, defining a new `Model` class derived from `BaseModel` and implementing [5] the set of methods detailed below:

- `__init__()`: Instantiates a model. Keyword arguments can be used to gather model specific options from the configuration file.
- `init_params()`: Initializes the layers and their weights.
- `build()`: Defines the computation graph for training.
- `build_sampler()`: Defines the computation graph for beam-search. This is similar to `build()` except two additional Theano functions.
- `load_valid_data()`: Loads the validation data for perplexity computation.
- `load_data()`: Loads the training data.

[5]The NMT architecture defined in `attention.py` can generally be used as a skeleton code when developing new architectures.

2.5. Building Blocks

Initialization Weight initialization is governed by the `weight_init` option and supports Xavier (Glorot and Bengio, 2010), He (He et al., 2015), orthogonal (Saxe et al., 2013) and random normal initializations.

Regularization An inverse-mode (the magnitudes are scaled during training instead of testing) dropout (Srivastava et al., 2014) can be applied over any tensor. L_2 weight regularization with a scalar factor given by `decay_c` option is also provided.

Option	Value	Description
`[training]` options		
`init`	**`None`**`/<.npz file>`	Pretrained checkpoint to initialize the weights.
`device_id`	**`auto`**`/cpu/gpu<int>`	Select training device automatically or manually.
`seed`	`1234`	The seed for Theano and NumPy RNGs.
`clip_c`	`5.0`	Gradient norm clipping threshold.
`decay_c`	`0.0`	L_2 regularization factor.
`patience`	`10`	Early-stopping patience.
`patience_delta`	`0.0`	Absolute difference of early-stopping metric that will be taken into account as an improvement.
`max_epochs`	`100`	Maximum number of epochs for training.
`max_iteration`	`1e6`	Maximum number of updates for training.
`valid_metric`	**`bleu`**`/meteor/px`	Validation metric(s) (separated by comma) to be printed, first being the early-stopping metric.
`valid_start`	`1`	Start validation after this number of epochs finished.
`valid_freq`	`0`	`0` means validations occur at end of epochs while an explicit `<int>` defines the period in terms of updates.
`valid_njobs`	`16`	Number of CPUs to use during validation beam-search.
`valid_beam`	`12`	The size of the beam during validation beam-search.
`valid_save_hyp`	**`False`**`/True`	Dumps validation hypotheses to separate text files.
`disp_freq`	`10`	The frequency of logging in terms of updates.
`save_best_n`	`4`	Save 4 best models on-disk based on validation metric for further ensembling.
`[model]` options		
`weight_init`	**`xavier`**`/he/<float>`	Weight initialization method or a `<float>` to define the scale of random normal distribution.
`batch_size`	`32`	Mini-batch size for training.
`optimizer`	**`adam`**`/adadelta/sgd/rmsprop`	Stochastic optimizer to use for training.
`lrate`	**`None`**`/<float>`	If given, overrides the optimizer default defined in `nmtpy/optimizers.py`.

*Table 2. Description of options and their default values: when the number of possible values is finite, the default is written in **bold**.*

Layers Feed-forward layer, highway layer (Srivastava et al., 2015), Gated Recurrent Unit (GRU) (Chung et al., 2014) Conditional GRU (CGRU) (Firat and Cho, 2016) and Multimodal CGRU (Caglayan et al., 2016a,b) are currently available for architecture design. Layer normalization (Ba et al., 2016), a method that adaptively learns to scale and shift the incoming activations of a neuron is available for GRU and CGRU blocks.

Iteration Parallel and monolingual text iterators with compressed (.gz, .bz2, .xz) file support are available under the names `TextIterator` and `BiTextIterator`. Additionally, the multimodal `WMTIterator` allows using image features and source/target sentences at the same time for multimodal NMT (Section 3.3). An efficient target length based batch sorting is available with the option `shuffle_mode:trglen`.

Training *nmtpy* provides Theano implementations of stochastic gradient descent (SGD) and its adaptive variants RMSProp (Tieleman and Hinton, 2012), Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2014) to optimize the weights of the trained network. A preliminary support for gradient noise (Neelakantan et al., 2015) is available for Adam. Gradient norm clipping (Pascanu et al., 2013) is enabled by default with a threshold of 5 to avoid exploding gradients. Although the provided architectures all use the cross-entropy objective by their nature, any arbitrary differentiable objective function can be used since the training loop is agnostic to the architecture being trained.

Post-processing All decoded translations will be post-processed if `filter` option is given in the configuration file. This is useful in the case where one would like to compute automatic metrics on surface forms instead of segmented. Currently available filters are *bpe* and *compound* for cleaning subword BPE (Sennrich et al., 2016) and German compound-splitting (Sennrich and Haddow, 2015) respectively.

Metrics *nmt-train* performs a patience based early-stopping using either validation perplexity or one of the automatic metric wrappers i.e. BLEU (Papineni et al., 2002) or METEOR (Lavie and Agarwal, 2007). These metrics are also available for *nmt-translate* to immediately score the produced hypotheses. Other metrics can be easily added and made available as early-stopping metrics.

3. Architectures

3.1. Neural Machine Translation (NMT)

The NMT architecture (**attention**) is based on *dl4mt-tutorial* which differs from Bahdanau et al. (2014) in the following major aspects:

- The decoder is CGRU (Firat and Cho, 2016) which consists of two GRU interleaved with attention mechanism,
- The hidden state of the decoder is initialized with a non-linear transformation applied to *mean* bi-directional encoder state instead of *last* one,
- Maxout (Goodfellow et al., 2013) layer before the softmax operation is removed.

Option	Value(s) (default)	Description
`init_cgru`	zero (text)	Initializes CGRU with zero instead of mean encoder state (García-Martínez et al., 2017).
`tied_emb`	2way/3way (False)	Allows 2way and 3way sharing of embeddings in the network (Inan et al., 2016; Press and Wolf, 2016).
`shuffle_mode`	simple (trglen)	Switch between simple and target-length ordered shuffling.
`layer_norm`	bool (False)	Enable/disable layer normalization for GRU encoder.
`simple_output`	bool (False)	Condition target probability only on decoder's hidden state (García-Martínez et al., 2017).
`n_enc_layers`	int (1)	Number of unidirectional encoders to stack on top of the bi-directional encoder.
`emb_dropout`	float (0)	Rate of dropout applied on source embeddings.
`ctx_dropout`	float (0)	Rate of dropout applied on source encoder states.
`out_dropout`	float (0)	Rate of dropout applied on pre-softmax activations.

Table 3. Description of configuration options for the NMT architecture.

The final NMT architecture offers many new options which are shortly explained in Table 3. We also provide a set of auxiliary tools which are useful for pre-processing and post-training tasks (Table 4).

Tool	Description
`nmt-bpe-*`	Clone of subword utilities for BPE processing (Sennrich et al., 2016).
`nmt-build-dict`	Generates `.pkl` vocabulary files from corpora prior to training.
`nmt-rescore`	Rescores n-best hypotheses with single/ensemble of models on GPU.
`nmt-coco-metrics`	Computes several metrics using MSCOCO evaluation tools (Chen et al., 2015).
`nmt-extract`	Extracts and saves weights from a trained model instance.

Table 4. Brief descriptions of helper NMT tools.

3.2. Factored NMT (FNMT)

Factored NMT (FNMT) is an extension of NMT which generates two output symbols (García-Martínez et al., 2016). In contrast to multi-task architectures, FNMT outputs share the same recurrence and output symbols are generated in a synchronous fashion. Two variants which differ in how they handle the output layer are currently available: (`attention_factors`) where the lemma and factor embeddings are concatenated to form a single feedback embedding and (`attention_factors_seplogits`) where the output path for lemmas and factors are kept separate with different pre-softmax transformations applied for specialization.

3.3. Multimodal NMT (MNMT)

We provide several multimodal architectures where the probability of a target word is estimated given source sentence representations and visual features: (1) Fusion architectures (Caglayan et al., 2016a,b) extend monomodal CGRU into a multimodal one where a multimodal attention is applied over textual and visual features, (2) MNMT architectures based on global features make use of fixed-width visual features to ground NMT with visual informations (Caglayan et al., 2017).

3.4. Other

- A GRU-based reimplementation (**img2txt**) of *Show, Attend and Tell* image captioning architecture (Xu et al., 2015),
- A GRU-based language model architecture (**rnnlm**) to train recurrent language models. *nmt-test-lm* is the inference utility for perplexity computation of a corpus using a trained checkpoint.

4. Results

System	MMT	Test2017 Meteor (Rank)
NMT	En→De	53.8 (#3)
MNMT	En→De	54.0 (#1)
NMT	En→Fr	70.1 (#4)
MNMT	En→Fr	72.1 (#1)
System	**News**	**Test2017 BLEU**
NMT-UEDIN (Winner)	En→Tr	16.5
NMT-Ours (Post-deadline)	En→Tr	18.1
FNMT	En→Lv	16.2
FNMT	En→Cs	19.9

Table 5. Ensembling scores for LIUM's WMT17 MMT and News Translation submissions.

System	Test2017 BLEU	Test2017 METEOR
Nmtpy	30.8 ± 1.0	51.6 ± 0.5
Nematus	31.6	50.6

Table 6. Mean/std. deviation of 5 Nmtpy runs vs 1 Nematus run for WMT17 MMT En→De.

We present our submitted *nmtpy* systems for Multimodal Translation (MMT) and News Translation tasks of WMT17 (Table 5). For MMT, state-of-the-art results are obtained by our systems (Caglayan et al., 2017)[6] in both En→De and En→Fr tracks (Elliott et al., 2017). In the context of news translation task, our post-deadline En→Tr NMT system (García-Martínez et al., 2017) surpassed the official winner by 1.6 BLEU.

We also trained a monomodal NMT for WMT17 MMT En→De track with Nematus using hyper-parameters very similar to our submitted NMT architecture and found that the results are comparable for BLEU and slightly better for *nmtpy* in terms of METEOR (Table 6).

5. Conclusion

We have presented *nmtpy*, an open-source sequence-to-sequence framework based on *dl4mt-tutorial* and refined in many ways to ease the task of integrating new architectures. The toolkit has been internally used in our team for tasks ranging from monomodal, multimodal and factored NMT to image captioning and language modeling to achieve top-ranked campaign results and state-of-the-art performance.

Acknowledgements

This work was supported by the French National Research Agency (ANR) through the CHIST-ERA M2CR project, under the contract number ANR-15-CHR2-0006-01[7].

Bibliography

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL `http://arxiv.org/abs/1607.06450`.

[6]`http://github.com/lium-lst/wmt17-mmt`

[7]`http://m2cr.univ-lemans.fr`

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

Caglayan, Ozan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. Does Multimodality Help Human and Machine for Translation and Image Captioning? In *Proceedings of the First Conference on Machine Translation*, pages 627–633, Berlin, Germany, August 2016a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2358.pdf.

Caglayan, Ozan, Loïc Barrault, and Fethi Bougares. Multimodal Attention for Neural Machine Translation. *arXiv preprint arXiv:1609.03976*, 2016b. URL http://arxiv.org/abs/1609.03976.

Caglayan, Ozan, Walid Aransa, Adrien Bardet, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, Marc Masana, Luis Herranz, and Joost van de Weijer. LIUM-CVC Submissions for WMT17 Multimodal Translation Task. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, September 2017.

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 preprint arXiv:1504.00325*, 2015.

Chung, Junyoung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Elliott, Desmond, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, September 2017.

Firat, Orhan and Kyunghyun Cho. Conditional Gated Recurrent Unit with Attention Mechanism. github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf, 2016.

García-Martínez, Mercedes, Loïc Barrault, and Fethi Bougares. Factored Neural Machine Translation Architectures. In *Proceedings of the International Workshop on Spoken Language Translation*, IWSLT'16, Seattle, USA, 2016. URL http://workshop2016.iwslt.org/downloads/IWSLT_2016_paper_2.pdf.

García-Martínez, Mercedes, Ozan Caglayan, Walid Aransa, Adrien Bardet, Fethi Bougares, and Loïc Barrault. LIUM Machine Translation Systems for WMT17 News Translation Task. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, September 2017.

Glorot, Xavier and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR, 13–15 May 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, Ian, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout Networks. In Dasgupta, Sanjoy and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine*

Learning Research, pages 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/goodfellow13.html.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1026–1034. IEEE, 2015.

Helcl, Jindřich and Jindřich Libovický. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17, 2017. ISSN 0032-6585. doi: 10.1515/pralin-2017-0001. URL http://ufal.mff.cuni.cz/pbml/107/art-helcl-libovicky.pdf.

Inan, Hakan, Khashayar Khosravi, and Richard Socher. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *arXiv preprint arXiv:1611.01462*, 2016.

Kingma, Diederik and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL http://arxiv.org/abs/1412.6980.

Klein, G., Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*, 2017.

Lavie, Alon and Abhaya Agarwal. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626389.

Neelakantan, Arvind, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015. URL http://arxiv.org/abs/1511.06807.

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*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043083.

Press, Ofir and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

Saxe, Andrew M, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Sennrich, Rico and Barry Haddow. A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 114–121. Association for Computational Linguistics, 2015.

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)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1162.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch-Mayne, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Miceli Barone, Jozef Mokry, and Maria Nadejde. *Nematus: a Toolkit for Neural Machine Translation*, pages 65–68. Association for Computational Linguistics (ACL), 4 2017. ISBN 978-1-945626-34-0.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016. URL http://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), 2012.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings, 2015. URL http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Zeiler, Matthew D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Appendix A: Example NMT Configuration

```
# Options in this section are consumed by nmt-train
[training]
model_type: attention   # Model type without .py
patience: 20            # early-stopping patience
valid_freq: 1000        # Compute metrics each 1000 updates
valid_metric: meteor    # Use meteor during validations
valid_start: 2          # Start validations after 2nd epoch
valid_beam: 3           # Decode with beam size 3
valid_njobs: 16         # Use 16 processes for beam-search
valid_save_hyp: True    # Save validation hypotheses
decay_c: 1e-5           # L2 regularization factor
clip_c: 5               # Gradient clip threshold
seed: 1235              # Seed for numpy and Theano RNG
save_best_n: 2          # Keep 2 best models on-disk
device_id: auto         # Pick 1st available GPU
max_epochs: 100

# Options in this section are passed to model instance
[model]
tied_emb: 2way          # weight-tying mode (False,2way,3way)
layer_norm: True        # layer norm in GRU encoder
shuffle_mode: trglen    # Shuffled/length-ordered batches
filter: bpe             # post-processing filter(s)
n_words_src: 0          # limit src vocab if > 0
n_words_trg: 0          # limit trg vocab if > 0
save_path: ~/models     # Where to store checkpoints
rnn_dim: 100            # Encoder and decoder RNN dim
embedding_dim: 100      # All embedding dim
weight_init: xavier
batch_size: 32
optimizer: adam
lrate: 0.0004
emb_dropout: 0.2        # Set dropout rates
ctx_dropout: 0.4
out_dropout: 0.4

# Vocabulary paths produced by nmt-build-dict
[model.dicts]
src: ~/data/train.norm.max50.tok.lc.bpe.en.pkl
trg: ~/data/train.norm.max50.tok.lc.bpe.de.pkl

# Training and validation data
[model.data]
train_src     : ~/data/train.norm.max50.tok.lc.bpe.en
train_trg     : ~/data/train.norm.max50.tok.lc.bpe.de
valid_src     : ~/data/val.norm.tok.lc.bpe.en
valid_trg     : ~/data/val.norm.tok.lc.bpe.de # BPE refs for validation perplexity
valid_trg_orig: ~/data/val.norm.tok.lc.de     # non-BPE refs for correct metric computation
```

Appendix B: Installation

nmtpy requires a Python 3 environment with NumPy and Theano v0.9 installed. A Java runtime (`java` should be in the `PATH`) is also needed by the METEOR implementation. You can run the below commands in the order they are given to install *nmtpy* into your Python environment:

```
# 1. Clone the repository
$ git clone https://github.com/lium-lst/nmtpy.git

# 2. Download METEOR paraphrase data files
$ cd nmtpy; scripts/get-meteor-data.sh

# 3. Install nmtpy
$ python setup.py install
```

Note that once you installed *nmtpy* with `python setup.py install`, any modifications to the source tree will not be visible until *nmtpy* is reinstalled. If you would like to avoid this because you are constantly modifying the source code (for adding new architectures, iterators, features), you can replace the last command above by `python setup.py develop`. This tells the Python interpreter to directly use *nmtpy* from the GIT folder. The final alternative is to copy `scripts/snaprun` into your `$PATH`, modify it to point to your GIT folder and launch training using it as in below:

```
$ which snaprun
/usr/local/bin/snaprun

# Creates a snapshot of nmtpy under /tmp and uses it
$ snaprun nmt-train -c wmt17-en-de.conf
```

Performance In order to get the best speed in terms of training and beam-search, we recommend using a recent version of CUDA, CuDNN and a NumPy linked against Intel MKL[8] or OpenBLAS.

Address for correspondence:
Ozan Caglayan
`ozancag@gmail.com`
Laboratoire d'Informatique de l'Université du Maine (LIUM)
Avenue Laënnec 72085
Le Mans, France

[8]Anaconda Python distribution is a good option which already ships an MKL-enabled NumPy.

The Prague Bulletin of Mathematical Linguistics
NUMBER 109 OCTOBER 2017 29–38

Parallelization of Neural Network Training for NLP with Hogwild!

Valentin Deyringer,[a b] Alexander Fraser,[a] Helmut Schmid,[a] Tsuyoshi Okita[a]

[a] Centrum für Informations- und Sprachverarbeitung, LMU München
[b] Gini GmbH, München

Abstract

Neural Networks are prevalent in todays NLP research. Despite their success for different tasks, training time is relatively long. We use Hogwild! to counteract this phenomenon and show that it is a suitable method to speed up training Neural Networks of different architectures and complexity. For POS tagging and translation we report considerable speedups of training, especially for the latter. We show that Hogwild! can be an important tool for training complex NLP architectures.

1. Introduction

Many novel Machine Translation (MT) systems make use of Neural Networks (NNs) of different structure. In contrast to other machine learning methods, NNs are able to learn the relevant characteristics of the data independently (Bengio et al., 2013) and thus do not rely on handcrafted features which in turn requires expert knowledge and extensive study of the data basis. Backed by growing amounts of data available and increasing computational power, NNs have achieved remarkable results in different disciplines (Goodfellow et al., 2016). NNs have also proven to perform very well for MT (Cho et al., 2014; Sutskever et al., 2014).

These promising results of adopting NNs for MT and especially their capability of capturing the semantics of phrases (Cho et al., 2014) led to the emergence of a new branch of research referred to as Neural Machine Translation (NMT). This approach addresses the problem of translation with techniques solely based on NNs. A comparably simple system has shown that an NMT system is able to reach near state-of-the-art results and even surpass a matured SMT system (Bahdanau et al., 2014).

Corresponding author: valentin@gini.net
Cite as: Valentin Deyringer, Alexander Fraser, Helmut Schmid, Tsuyoshi Okita. Parallelization of Neural Network Training for NLP with Hogwild!. The Prague Bulletin of Mathematical Linguistics No. 109, 2017, pp. 29–38.
doi: 10.1515/pralin-2017-0036.

A major drawback of NMT systems attenuating the positive findings is the long time needed to train the translation models. The most widely used gradient based optimization algorithms SGD, Adagrad (Duchi et al., 2011) Adadelta (Zeiler, 2012), Adam (Kingma and Ba, 2014) and RMSprop (Tieleman and Hinton, 2012) show good convergence properties for optimizing NNs and can be efficiently implemented by moving the underlying matrix operations to GPUs for heavy parallelization (e.g., with frameworks like *theano* (Bergstra et al., 2010) or *Tensorflow* (Abadi et al., 2016)). This approach obtains considerable speedups (Brown, 2014). There are several libraries for programming languages which offer a convenient interface for GPU programming in the context of NNs. Nowadays, almost all real world applications of bigger NN models involve computation on GPUs.

Dependent on quantity of training data and model size, which both generally have a positive effect on the resulting models quality when increased, training NMT systems reportedly still requires several days. Training times of 3 to 10 days are common (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2014). In consequence, other ways to speed up the training are desirable. Besides from using GPUs, a way to shorten training times is parallelization on a higher level. This is not a trivial task as all of the optimization algorithms mentioned earlier are inherently sequential procedures. Nevertheless, there are generally two distinct approaches to achieve such parallelism, namely *model parallelism* and *data parallelism*. These approaches do not restrict the application of GPUs for the underlying matrix calculations and allow making use of the combined strength of several GPUs in a cluster.

The method of model parallelism distributes different computations performed on the same data onto multiple processors. The results are then merged in an appropriate way by a master process which also handles communication between processors as they are dependent on the results computed by the other processors. This technique is well suited for NNs due to their structure and is successfully implemented for the training of NMT models in (Sutskever et al., 2014). However, the work in hand is not concerned with model parallel approaches.

Data parallelism pursues a different approach where the processors perform the same operation on different data. In terms of optimization of NNs, this means that the training data is divided among the processors while shared parameters of the network are updated according to a suitable schedule. Data parallel training of NNs is not a trivial task and the commonly used optimization algorithms for training NNs are inherently iterative. Nevertheless, there are approaches in a data parallel fashion that allow parallelization of NN optimization, one of which is Hogwild! (Niu et al., 2011).

Hogwild! is an instance of a data parallel approach where updates to the global parameters are applied without locks. In this work we will show that Hogwild! can be successfully applied to train NNs for NMT as well as for POS tagging. The main contribution of this work is the implementation of this algorithm for theano.[1]

The final results suggest that fitting NMT models with this asynchronous optimization technique has the potential to speed up the training process. It is found that Hogwild! is well suited for parallelized training of NMT models. As a secondary finding, an additional experiment shows that the same algorithms can be applied to NNs of various structures.

2. Approach

In SGD and descendant algorithms, updates are calculated with parameters estimated in the previous time step. Therefore these algorithms are sequential in nature. While basically applying the same update rule as standard SGD, in Hogwild!, separate updates for different batches of data are calculated on each working node based on parameters shared among all working nodes. These shared parameters are read and written to without any locks which usually are used to avoid simultaneous read/write operations on the same data in parallelized programs. As a result, the parameters possibly lack some updates computed on other processors that are yet to be applied and occasional overwrites may occur. However, assuming sparsity in the parameters updated for each training example, Niu et al. (2011) show that these downsides have negligible impact on the training procedure. With the results presented in Section 5, we demonstrate that this algorithm is also successfully applicable to NN training.

We implemented Hogwild! for the Theano framework using Python's multiprocessing module. After initializing the weights and defining the model's computational graph, several worker processes are spawned and local copies of the graph are compiled for each. This is necessary due to Theano functions not being thread safe. The subprocesses read batches of training data from a queue and when a new batch of data is processed, the globally shared variables are read and updates are calculated accordingly. These updates are then sent back to be applied to the shared parameters. In accordance with the update scheme of Hogwild! the shared parameters are read and written to without any locking. For more detail we refer the interested reader to our source code.

Especially in the case of using GPUs, data transfer to and from device memory may slow down training. However, in our experiments we did not find this to have a strong impact. Rather, due to Theano's GPU capabilities it is easy to utilize GPUs as working nodes and benefit from their strengths for matrix calculations.

[1]Our implementation of Hogwild! for Theano can be found at http://github.com/valentindey/async-train.

3. Related Work

Introducing the algorithm, Niu et al. (2011) compare Hogwild! to a version thereof with locking and the asynchronous optimization strategy presented by Langford et al. (2009) and demonstrate that Hogwild! obtains improved speed for several problems. The positive findings make it a natural choice for us to apply this algorithm to problems of NLP. However the examined tasks in the original paper only give little indication for applicability of Hogwild! for optimization of NNs as they are mostly used, especially in NLP.

The single machine C implementation of word2vec released as part of the work of Mikolov et al. (2013) also uses lock-free updates in the style of Hogwild!.[2] This method is clearly useful in this setting, as the problem typically is very sparse with large vocabulary sizes and only a few words affected at each update. Training word embeddings this way is based on a relatively simple NN architecture. We train more complex models with Hogwild! without such a clear notion of sparsity, and our approach allows us to use Hogwild! with flexibly defined complex NN architectures.

Feng et al. (2016) present an evaluation of different optimization algorithms on question answering tasks. Among other algorithms, implementations of EASGD/ EAMSGD (Zhang et al., 2015) and Downpour SGD (Dean et al., 2012) are evaluated. While showing promising results for parallelized gradient based optimization, their study lacks comparison with Hogwild! which we find is a suitable method for optimizing NNs.

Building on Hogwild!, Noel and Osindero (2014) introduce an optimization technique working on computing clusters like multiple CPU cores, multiple GPUs, or several machines. They implemented this for the Caffe framework (Jia et al., 2014) and show brief benchmarks on the MNIST (Lecun et al., 2009) and ImageNet (Deng et al., 2009) data sets, depicting promising results for applications using NNs. Inter alia, we take these findings as basis for porting Hogwild! to NLP problems.

The NMT systems marian[3] and OpenNMT[4] comprise implementations of asynchronous update strategies similar to Hogwild!. As marian is written in C++ and OpenNMT is built with the lua framework torch it is of interest to have a point of reference for a system implemented with Theano in python. Additionally, we are able to attain better speedup properties when increasing the number of used GPUs compared to the benchmarks listed on the website of marian.

[2]Their published results used the DistBelief framework (Dean et al., 2012) which follows a different parallelization paradigm since one of the goals is to overcome the constraint of having only small RAM in GPUs.

[3]https://marian-nmt.github.io

[4]http://opennmt.net

4. Tasks

POS Tagging Part of speech tagging is one of the most fundamental problems in NLP and can also be used to improve machine translation systems (Ueffing and Ney, 2003).

We study a German POS Tagging task using a self-defined NN model. Similar to Ling et al. (2015), our model represents words by the concatenation of a word embedding and a character-based word representation. The latter is computed with a bidirectional LSTM (BiLSTM) from the word's character sequence. Characters occurring only once and words occurring less than 10 times are replaced by a special symbol UNK. The forward/backward character LSTM processes the word suffix/prefix of length 10. The suffix/prefix is padded with padding symbols if the word length is below 10. The final states of the forward and backward LSTMs and the word embedding are concatenated. The resulting sequence of word representations is processed by a second BiLSTM whose forward and backward states are concatenated at each position. Each positional representation obtained in this way is linearly projected to an output layer with a softmax activation function over possible POS tags.

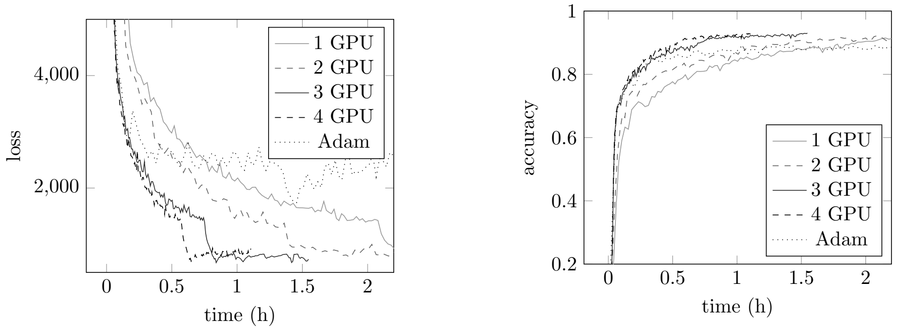
We use character embeddings of size 100, word embeddings of size 800, a character BiLSTM of size 400 for both directions and a deep word BiLSTM of size 800 for both directions with two layers of equal size. The training maximizes the log-likelihood of the correct tags. We decrease the initial learning rate of 0.03 by 0.0135 after each epoch. The character BiLSTMs are processed in parallel for all input words, but otherwise no batch processing is applied.

We trained our model on the German TIGER corpus (Brants et al., 2002) which is annotated with fine-grained POS tags that include additional annotations (e.g., number, gender, and case for nouns). We train on 40472 sentences and evaluate our models on 5,000 sentences held out from training.

Neural Machine Translation NMT systems working on the sentence level make use of the so-called encoder-decoder architecture which transforms input sentences into vector representations via an encoder RNN and decodes the target sentences with a decoder RNN (Sutskever et al., 2014; Cho et al., 2014).

The NMT model used for this work is based on the dl4mt material[5]. It uses gated recurrent units (GRUs) as introduced by Cho et al. (2014) with 1,000 units for both, the encoder and the decoder, and applies an attention mechanism (Bahdanau et al., 2014). All data is tokenized in a preprocessing step with the tokenizing script from Moses (Koehn et al., 2007). The 15,000 most common words of the source and target languages are mapped to embeddings of size 100. All other words are treated as unknown and mapped to a shared embedding. We ignore sentences longer than 50

[5]The original dl4mt code can be found at https://github.com/nyu-dl/dl4mt-tutorial and the code for our adapted version lives at https://github.com/valentindey/pnmt



(a) decrease of loss during training *(b) accuracy on data held out from training*

Figure 1: Training POS tagging models

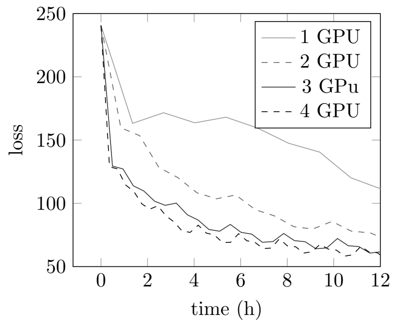
words. We use about 1.6 million sentences from the French/English Europarl corpus (Koehn, 2005) for training and an additional 10,000 sentences from the same corpus held out from training to monitor the training procedure (e.g. for early stopping). We maximize the log-likelihood of the training data with an initial learning rate of 0.1 that is not decreased throughout training. Gradients larger than 1.0 are clipped. For our NMT experiments, we use a batch size of 64.

5. Experimental Results

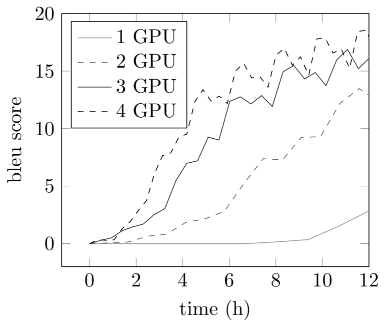
POS Tagging We trained the POS tagging model described in the previous section for 2 epochs on a machine running Ubuntu 16.04 equipped with Nvidia GeForce GTX 1080 units. As Figure 1 shows, this short training period is sufficient to show the performance gains through applying Hogwild!.

For comparison we also train the model with Adam on one GPU with the commonly used default hyperparameter values recommended by Kingma and Ba (2014).

Figure 1 shows the trajectories of training error and accuracy on the held-out data during training. There is a clear speedup when increasing the number of working nodes, and training with Hogwild! quickly becomes superior to training with Adam. The notches of training error in Figure 1a result from the decrease of learning rate after the first epoch. It is interesting to see that Adam fails to decrease the loss during training from a certain point on, probably oscillating around a local minimum while the models trained with Hogwild! decrease the loss further. The common strategy of initial training with an advanced optimization algorithm like Adam followed by SGD for fine tuning (e.g. Sercu et al., 2016) is thus no longer necessary when using



(a) decrease of loss for a subset of the Europarl training data



(b) BLEU score for the Europarl data held out from training

Figure 2: Training NMT models

number of GPUs	time (h)	train loss	BLEU score held out data
1	26.89	68.4441	14.42
2	16.56	64.5328	14.46
3	9.24	64.1065	14.32
4	6.96	64.2586	14.40

Table 1: performance of NMT models after 90,000 updates

Hogwild! and near optimal parameters are still found faster than when only using SGD.

Machine Translation The previous findings support the use of Hogwild! for the somewhat smaller problem of POS tagging where training is relatively fast. In the case of the more complex problem of NMT where training times of several days are common (Bahdanau et al., 2014; Sutskever et al., 2014; Cho et al., 2014) we see an even more marked improvement due to parallelization with Hogwild!.

The numbers in Table 1 exemplify the achieved speedup by listing the times needed for 90,000 updates and the respective model performance for different levels of concurrency. As expected, the time required to perform a fixed number of updates is decreasing approximately linearly dependent on the number of working nodes, i.e., GPUs. This indicates little to no influence of the negative side effects of Hogwild! discussed in Section 2. The models' performance in terms of BLEU scores is almost unchanged and we can train competitive models in substantially shorter time. Applying Hogwild! for this problem in our eyes shows the most gain as this kind of model usually takes a very long time to train. However Figure 2 also suggests that there is an upper bound to the gains through parallelization which is almost reached when working on four GPUs, the same as with the POS tagging model.

Summary of NLP Results With Hogwild! we can speed up the training for both of the problems we considered, including different kinds of models. We can also see that using more GPUs has less impact on the training progress with each increment, which indicates an upper bound for positive effects from increased concurrency. Using three or four GPUs works very well for appropriately sized models and complex problems.

6. Conclusion

We have shown that Hogwild! is useful for training NNs of different architectures faster. It is meaningful to train models on multiple GPUs and CPU cores. The adverse effects of Hogwild! discussed in Section 2 are at a negligible level for the stated problems and show that the algorithm is suitable for training NNs for NLP tasks. We find that running Hogwild! on three to four GPU devices gives viable results for POS tagging and NMT.

With the release of our source code, we provide the means to easily use Hogwild! for other systems implemented in Theano. The seq2seq module in Tensorflow provides the GPU parallelization in a layerwise manner automatically when the LSTM consists of multiple layers. Hogwild! can be deployed on top of this setting easily as well.

Recently typical GPU environments have changed drastically. The Nvidia PASCAL architecture provides bigger graphics memory at a cheaper price point. This means that setups with four GPUs (or even eight or sixteen) are becoming widely accessible, an interesting contrast with massive CPU parallelization (Dean et al., 2012). In the computer vision community, parallel GPU architectures like the server we used are heavily used, while in the NLP community they are rare. Our results show that the NLP community should more strongly consider training with multiple parallel GPUs.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644402 (HimL). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 640550).

Bibliography

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Bergstra, James, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A CPU and GPU math compiler in Python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168, 2002.

Brown, Larry. Accelerate Machine Learning with the cuDNN Deep Neural Network Library, 2014. URL `https://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library`. [Online; accessed 2016-07-14].

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Duchi, John, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Feng, Minwei, Bing Xiang, and Bowen Zhou. Distributed deep learning for question answering. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2413–2416. ACM, 2016.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

Kingma, Diederik and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

Langford, John, Alex J Smola, and Martin Zinkevich. Slow learners are fast. *Advances in Neural Information Processing Systems*, 22:2331–2339, 2009.

Lecun, Yann, Corinna Cortes, and Christopher JC Burges. The MNIST database of handwritten digits, 2009. 2009. URL `http://yann.lecun.com/exdb/mnist`.

Ling, Wang, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Niu, Feng, Benjamin Recht, Christopher Re, and Stephen Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

Noel, Cyprien and Simon Osindero. Dogwild! - Distributed Hogwild for CPU & GPU. In *NIPS Workshop on Distributed Machine Learning and Matrix Computations*, 2014.

Sercu, Tom, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for LVCSR. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4955–4959. IEEE, 2016.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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), 2012.

Ueffing, Nicola and Hermann Ney. Using pos information for statistical machine translation into morphologically rich languages. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 347–354. Association for Computational Linguistics, 2003.

Zeiler, Matthew D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, Sixin, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging SGD. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.

Address for correspondence:
Valentin Deyringer
`valentin@gini.net`
Gini GmbH
Prannerstraße 10, D-80333 Munich, Germany

PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 109 OCTOBER 2017 39–50

Visualizing Neural Machine Translation Attention and Confidence

Matīss Rikters,[a] Mark Fishel,[b] Ondřej Bojar[c]

[a] Faculty of Computing, University of Latvia
[b] Institute of Computer Science, University of Tartu
[c] Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

In this article, we describe a tool for visualizing the output and attention weights of neural machine translation systems and for estimating confidence about the output based on the attention.

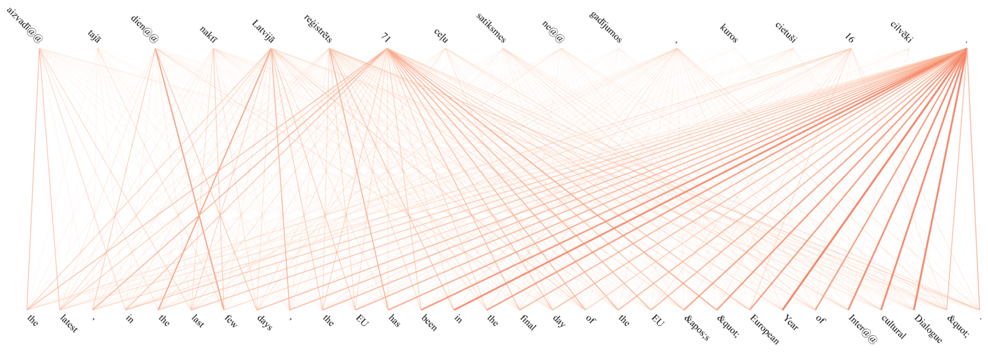
Our aim is to help researchers and developers better understand the behaviour of their NMT systems without the need for any reference translations. Our tool includes command line and web-based interfaces that allow to systematically evaluate translation outputs from various engines and experiments. We also present a web demo of our tool with examples of good and bad translations: `http://ej.uz/nmt-attention`.

1. Introduction

The world of machine translation (MT) is in transition between the well-recognized statistical MT (SMT, Koehn, 2009) and the new and exciting neural MT (NMT, e.g. Bahdanau et al., 2014). While the systems themselves are slowly being replaced, the necessities behind analyzing them remain the same, as do the tools built mostly for the older approaches.

In this paper, we introduce a translation inspection tool that specifically targets NMT output. The tool uses the attention weights corresponding to specific token pairs during the decoding process, by turning them into one of several visual representations that can help humans better understand how the output translations were produced. The tool also uses the attention information to estimate the confidence in

Corresponding author: matiss@lielakeda.lv
Cite as: Matīss Rikters, Mark Fishel, Ondřej Bojar. Visualizing Neural Machine Translation Attention and Confidence. The Prague Bulletin of Mathematical Linguistics No. 109, 2017, pp. 39–50.
doi: 10.1515/pralin-2017-0037.



Source:	Aizvadītajā diennaktī Latvijā reģistrēts 71 ceļu satiksmes negadījumos, kuros cietuši 16 cilvēki.
Hypothesis:	The latest, in the last few days, the EU has been in the final day of the EU's "European Year of Intercultural Dialogue".
Reference:	71 traffic accidents in which 16 persons were injured have happened in Latvia during the last 24 hours.

Figure 1. A Latvian to English neural translation output that has no relation to the input. The weak connection is obvious from the visualized attention weights, even without knowing the source and target languages or seeing the input or output texts. Confidence: ***18.11%***; *CDP: 44.49%; APout: 67.41%; APin: 79.58%.*

translation which allows to distinguish acceptable outputs from completely unreliable ones, no reference translations are required.

The paper is structured as follows: Section 2 summarizes related work on tools for inspecting translation outputs and alignments. Section 3 describes the tool from the users' point of view, covering the web-based and command-line visualizations and the confidence score for better navigation. Section 4 provides a look into the back-end of the system. Finally, conclusions and future work directions are in Section 5.

2. Related Work

Zeman et al. (2011) describe Addicter—a set of command-line and simple web-based tools that can be useful for inspecting automatic translations and finding systematic errors among them. One of the tools in Addicter, *alitextview.pl*, is designed to convert SMT alignments from the typical alignment pair format (*source_token_id – target_token_id*) to a table representation, making it more human-readable. Our command-line interface took much inspiration from this work while adapting to the specifics of the NMT counterpart of alignments.

Madnani (2011) introduces iBLEU—a web-based tool for visualizing BLEU (Papineni et al., 2002) scores. Unlike alignments between the source and the hypothesis, the calculation of BLEU requires a reference translation to which the hypothesis will be compared. On top of that, iBLEU also allows to add another file with hypotheses from another MT system for a direct comparison. Given these inputs, the tool highlights the differences between the translations and reference material. It also enables easy navigation through the set of sentences by representing the BLEU score of each sentence in a clickable bar chart. A quick jump to a specific sentence is possible by entering its number. The clickable chart and jumps seemed most desirable features for us, so we added similar capabilities to the web version of our tool.

Klejch et al. (2015) developed MT-ComparEval—a web-based translation visualization tool that seems to build upon iBLEU by adding many more fine-grained features. It also allows to compare differences between translations and references, other translations and the source input. The main differences are that (1) MT-ComparEval stores all imported data as experiments for viewing at any time, where iBLEU forgets everything upon a page refresh; (2) for each of these experiments, one can add output from multiple systems (iBLEU can cope with only 2); (3) MT-ComparEval displays additional scores (precision, recall, F-measure); and (4) it shows various detailed sentence and n-gram level statistics with configurable highlighting of the differences. A noticeable shortcoming is that one cannot jump to a specific sentence in the set. While ordering by sentence ID is possible, to view the 1000th of 2000 one would have to scroll through the first 999.

Nematus (Sennrich et al., 2017) includes a set of utilities for visualizing NMT attentions. The first one, *plot_heatmap.py* plots alignment matrices similar to the previously mentioned *alitextview.pl*, using Nematus output translations with alignments. The second tool, *visualize_probs.py* generates HTML for a web view that displays the output translation in a table with the background of each token shaded according to the attention weight. The final tool, consisting of *attention.js* and *attention_web.php*, connects source and target tokens with lines as thick as the corresponding attention weights between them. However there is no tool included to generate the latter visualization for an arbitrary sentence - it is given only in the form of one set example. This last tool was a strong inspiration for building our tool. We reused parts of its code in the web version of our visualization.

Neural Monkey (Helcl and Libovický, 2017) provides several visualization tools for checking the training process that include visualizing attention as soft alignments. It can generate matrices similar to the previously mentioned *alitextview.pl* for each sentence in the first validation batch during the training process. A few drawbacks of this method are that the images are (1) of a static size (the predefined maximum input length * maximum output length) - if sentences are longer, the attention image gets cut off, if shorter, bottom rows of the matrix (representing the input) are left black and columns (representing the output) on the far right side are filled with "phantom" attention; (2) no input and output words, tokens or subword units are displayed, only the matrix; (3) there is no option to generate visualizations for a test set outside the system training process.

3. The Tool from Users' Perspective

The main goals of our tool are to provide multiple ways of visualizing NMT attention alignments, as well as to make it easy to navigate larger data sets and find specific examples. To accomplish these goals, we implemented two main variations of our tool, a textual command line visualization and a web-based visualization. This chapter provides an insight into the features of both of them and suggestions as to when they can be useful.

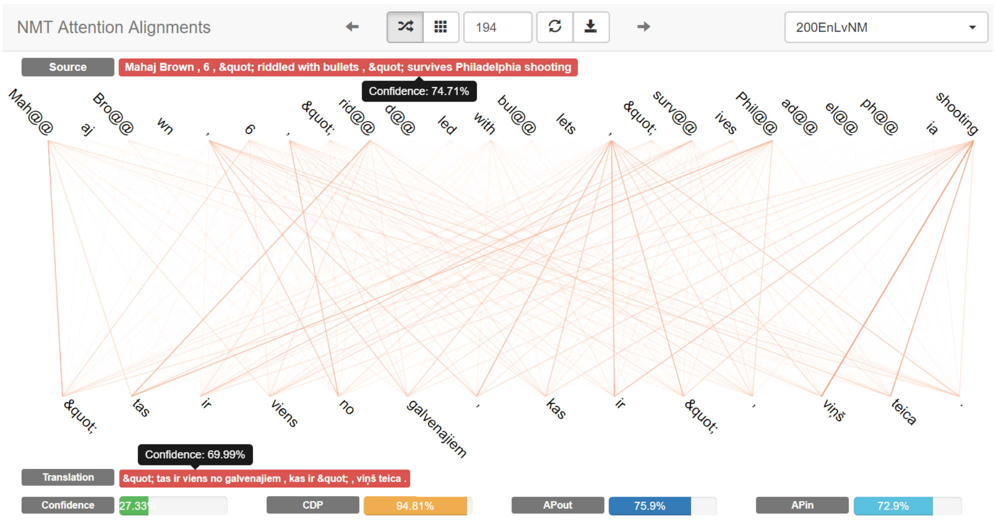
3.1. Web Browser Visualization

The web visualization is intended to provide an intuitive overview of one or multiple translated test sets. This is done by showing one sentence at a time, with navigation to other sentences by ID, length or multiple confidence measures. Switching between experiments (test sets) is also easy. For each individual sentence, four confidence metrics are shown, and a confidence score for each source and translated token (or subword unit). The tool also allows to export the alignment visualization of any selected sentence to a high-resolution PNG file with one click.

The essential part of the visualization is presented in the following way: source tokens (at the top) are connected to translated tokens (at the bottom) via orange lines, ranging from completely faint to very thick, as shown in Figures 2 and 3. A thicker line from a translated token to a source token means that the decoder paid more attention to that source token when generating the translation. Ideally, these lines should mostly be thick with some thinner ones in between. When they look chaotic, connecting everything to everything (Figure 2) or everything in the translation is connected to mostly a single token in the source,[1] that can be well an indication of an unsuccessful translation that will possibly have little or no relation with the source sentence. On the other hand, if all lines are thick, straight downwards, connected one-to-one (see the right part of Figure 3), that may point to nothing being translated at all.

Additionally, the matrix style visualization is also available in the web version as shown on the left part of Figure 3.

[1]Such tokens were called "garbage tokens" in IBM-style word-alignment methods (Och and Ney, 2000), and they were often rare words where the model had the option to attribute everything to them.


NMT Attention Alignments

194

200EnLvNM

Source: Mahaj Brown , 6 , " riddled with bullets , " survives Philadelphia shooting

Confidence: 74.71%

Confidence: 69.99%

Translation: &quot; tas ir viens no galvenajiem , kas ir " , viņš teica .

Confidence: 27.33%

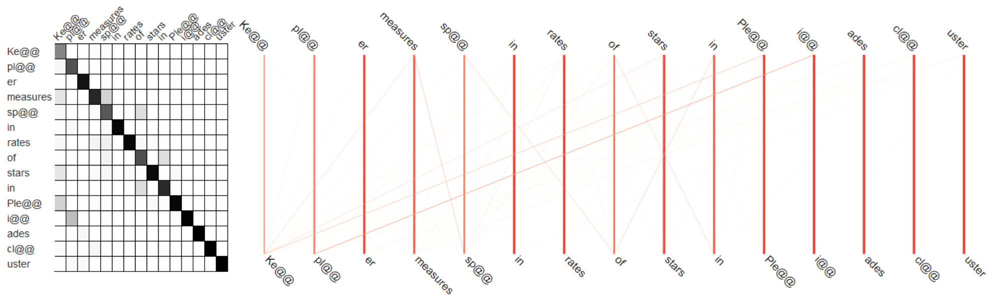
CDP: 94.81%

APout: 75.9%

APin: 72.9%

Source:	Mahaj Brown , 6 , ”riddled with bullets ,” survives Philadelphia shooting
Hypothesis:	”tas ir viens no galvenajiem , kas ir” , viņš teica.
Reference:	6 gadus vecais Mahajs Brauns ”ložu sacaurumots” izdzīvo apšaudē Filadelfijā.

*Figure 2. An example of a translated sentence that exhibits a low confidence score. Confidence: **27.33%**; CDP: 94.81%; APout: 75.9%; APin: 72.9%.*



Source: Kepler measures spin rates of stars in Pleiades cluster
Hypothesis: Kepler measures spin rates of stars in Pleiades cluster
Reference: Keplers izmēra zvaigžņu griešanās ātrumu Plejādes zvaigznājā.

Figure 3. An example of a translated sentence that exhibits a suspiciously high confidence score. The translation here is a verbatim rendition of the input. Matrix form visualization on the left, line form visualization on the right. Confidence: ***95.44%****; CDP: 100.0%; APout: 98.84%; APin: 98.85%.*

3.2. Confidence Scores

To aid in locating suspicious and potentially bad translations, we introduced a set of confidence metrics (more details in Section 4.1). For each sentence, the tool displays an overall confidence score, coverage deviation penalty, and input and output absentmindedness penalties. The overall confidence score is also shown for each source token, indicating the amount of confidence that the token has been used to generate a correct translation, as well as for each translated token, indicating the amount of confidence that it is a correct translation. All of these scores are represented in percentages from 0 to 100 and can be used to navigate through the test set (Figure 4), making it easy to quickly find very good or very bad translations among hundreds. The selected sentence is highlighted simultaneously across all navigation charts and each chart can be sorted in either direction or reset to the order by sentence ID.

3.3. Command Line Visualization

The command line visualization is available in three different formats: (1) using twenty-five different shades of gray as shown in Figure 5; (2) using five gradually shaded Unicode block elements as shown in Figure 6; and (3) using nine gradually filled Unicode block elements. Each sentence is output via a graphical matrix, where rows represent the source input tokens or subword units and columns representing the target side. The corresponding tokens are printed out on the bottom (target) or far right side (source) of the matrix. Unlike the authors of *alitextview.pl*, we chose

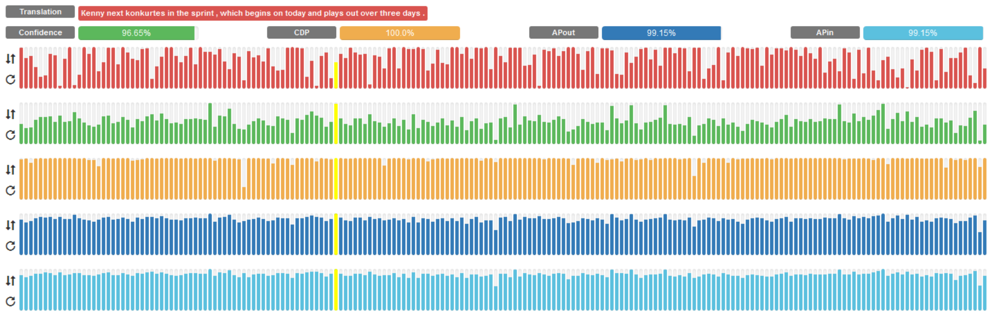
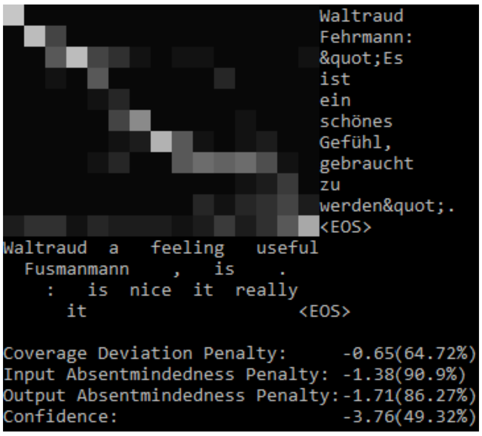

Translation
Kenny next konkurtes in the sprint , which begins on today and plays out over three days .
Confidence
96.65%
CDP
100.0%
APout
99.15%
APin
99.15%

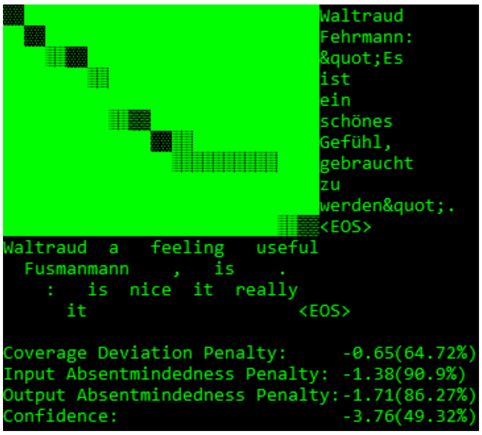
Figure 4. Navigation charts allow to jump to a sentence based on its length in characters (red), confidence (green), coverage deviation penalty (dark yellow), absentmindedness penalty for input (dark blue) and output (light blue). The currently active sentence is highlighted in bright yellow. All charts are sortable and scrollable for a better user experience.



```
Waltraud
Fehrmann:
&quot;Es
ist
ein
schönes
Gefühl,
gebraucht
zu
werden&quot;.
<EOS>
Waltraud  a  feeling  useful
  Fusmanmann   ,  is    .
    :  is  nice  it  really
      it                <EOS>

Coverage Deviation Penalty:     -0.65(64.72%)
Input Absentmindedness Penalty: -1.38(90.9%)
Output Absentmindedness Penalty:-1.71(86.27%)
Confidence:                     -3.76(49.32%)
```

Figure 5. Visualization in the command line, using twenty-five different tones of gray.



```
Waltraud
Fehrmann:
&quot;Es
ist
ein
schönes
Gefühl,
gebraucht
zu
werden&quot;.
<EOS>
Waltraud  a  feeling  useful
  Fusmanmann   ,  is    .
    :  is  nice  it  really
      it                <EOS>

Coverage Deviation Penalty:     -0.65(64.72%)
Input Absentmindedness Penalty: -1.38(90.9%)
Output Absentmindedness Penalty:-1.71(86.27%)
Confidence:                     -3.76(49.32%)
```

Figure 6. Visualization in the command line, using five differently shaded block elements.

to represent the source tokens on the right, so that the graphical matrix starts at the beginning of the line for each sentence. After each sentence, one empty line is printed.

One obvious use case for the command-line visualization is to directly compare alignments of NMT attention with the ones produced by SMT. This type of visualization is also the fastest, therefore it can be used to quickly check alignments for a specific sentence. Fixed-width Unicode fonts can be used in almost all text editors, so redirecting output in the *block* mode to a text file to share with others is also a useful application. However, to view the *color* version from a text file, it needs to be interpreted as xterm color sequences, e.g. using "`less -R`" in a Linux terminal.

4. System Description

The visualization tool is developed in Python and PHP. It is published in a GitHub repository[2] and open-sourced with the MIT License.

Both visualizations can be run directly from the command line. The web version is capable of launching on a local machine without the requirement for a dedicated web server.

4.1. Scoring Attention

This section provides details about how the previously mentioned confidence scores are calculated and outlines what is needed to make good use each option.

The basis of our scoring methods was influenced by Wu et al. (2016), who defined a coverage penalty for punishing translations that do not pay enough attention to input tokens:

$$CP = \beta \sum_{j} \log\left(\min\left(\sum_{i} \alpha_{ji}, 1.0\right)\right), \tag{1}$$

where CP is the coverage penalty, i is the output token index, j is the input token index and β is used to control the influence of the metric. To complement that, we introduce a set of our own metrics:

- **Coverage Deviation Penalty** (CDP) penalizes attention deficiency and excessive attention per input token.
- **Absentmindedness Penalties** ($AP_{out,\ in}$) penalize output tokens that pay attention to too many input tokens, or input tokens that produce too many output tokens.
- **Confidence** is the sum of the three metrics – CDP, AP_{in} and AP_{out}.

Coverage Deviation Penalty

Unlike CP, CDP penalizes not just attention deficiency but also excessive attention per input token. The aim is to penalize the sum of attentions per input token for going

[2]NMT Attention Alignment Visualizations: `https://github.com/M4t1ss/SoftAlignments`

too far from 1.0, so that tokens with the total attention of 1.0 get a score of 0.0 on the logarithmic scale, while tokens with less attention (like 0.13) or more attention (like 3.7) get lower values. We thus define the coverage deviation penalty:

$$\mathrm{CDP} = -\frac{1}{J}\sum_{j}\log\left(1+\left(1-\sum_{i}\alpha_{ji}\right)^{2}\right). \tag{2}$$

The metric is on a logarithmic scale, and it is normalized by the length J of the input sentence in order to avoid assigning higher scores to shorter sentences.

Absentmindedness Penalties

To target scattered attention per output token, we introduce an absentmindedness penalty:

$$\mathrm{AP}_{out} = -\frac{1}{I}\sum_{i}\sum_{j}\alpha_{ji}\cdot\log\alpha_{ji}. \tag{3}$$

It evaluates the dispersion via the entropy of the predicted attention distribution, resulting in values from 1.0 for the lowest entropy to 0.0 for the highest. The values are again on the log-scale and normalized by the source sentence length I.

The absentmindedness penalty can also be applied to the input tokens after normalizing the distribution of attention per input token:

$$\mathrm{AP}_{in} = -\frac{1}{I}\sum_{j}\sum_{i}\alpha_{ij}\cdot\log\alpha_{ij}. \tag{4}$$

The final confidence score sums up all three above mentioned metrics:

$$confidence = CDP + AP_{out} + AP_{in}. \tag{5}$$

For visualization purposes each of the scores needed to be set on the same scale of 0-100%. To achieve that, we applied

$$\text{percentage} = e^{-C(X^{2})}, \tag{6}$$

where X is the score to convert and C is a constant of either 1 for CDP or 0.05 for the other scores ($\mathrm{AP}_{\mathrm{out}}$, $\mathrm{AP}_{\mathrm{in}}$, confidence). Other constants were also tested, but these specific ones seemed to best fit data from our test sets, by displaying the percentage values across the whole range.

4.2. System Architecture

The code can be divided into two logical parts - 1) processing input data and generating output data and 2) displaying and navigating the generated output data in a web browser. The former part is written in Python and handles all input data, generates output data, displays the command line visualization or launches a temporary web server for the web browser visualization. Each time a web visualization is launched, a new folder is created within */web/data* where all necessary output data files are stored, a temporary PHP web server is launched on *127.0.0.1:47155*, and the address is opened as a new tab in the default web browser. After stopping the script all data remains in the */web/data* and can be accessed later as well.

The latter part is responsible for everything that is shown in the browser. It mainly consists of PHP, HTML and JavaScript code that facilitates quick navigation between sentences even in larger data files, as well as navigation charts and sorting, visualization export to image files and a responsive user interface. If necessary, this part can be used as a stand-alone website for displaying and interacting with pre-generated results.

4.3. Requirements and Usage

The requirements are as follows:

- Python (2 or 3) and NumPy,
- PHP 5.4 or newer (for web visualization),
- Nematus or Neural Monkey (for training NMT systems),
- Nematus, AmuNMT[3] (Junczys-Dowmunt et al., 2016) or Neural Monkey (for translating and extracting attention data)
  - Or any NMT framework that can output an attention matrix for each translation (may require format conversion) .

To use the tool, first translate a set of sentences using a supported NMT framework with the option of saving alignments[4] switched on. The sources combined with the resulting translations and attention matrices can then be used as input for the *process_alignments.py* script. Depending on the selected output type, alignments will either be displayed in the terminal or a new tab will be opened in the default web browser. Example input files from each supported NMT framework are provided along with commands to run them.

5. Conclusions

In this paper, we described our tool for visualizing attention alignments generated by neural machine translation systems and for estimating confidence of the translation. The tool aims to help researchers better understand how their systems perform by enabling to quickly locate better and worse translations in a bigger test set. Compared to other similar tools, ours relies on the confidence scores and does not require

[3]Barvins/amunmt (forked from marian-nmt/marian): `https://github.com/barvins/amunmt`

[4]How to get alignment files from NMT systems: `https://github.com/M4t1ss/SoftAlignments`

reference translations to facilitate this easier navigation. This allows to integrate it, for example, in an NMT system with a web interface, providing users with an explanation for the result of a specific translation.

In the future, we plan to integrate a part of this tool into one public NMT system, Neurotolge.[5] We will also extend the out-of-the-box support to other popular NMT frameworks like OpenNMT[6] or tensor2tensor.[7]

Acknowledgements

A part of this research was supported by the ICT COST Action IC1207 *ParseME: Parsing and multi-word expressions. Towards linguistic precision and computational efficiency in natural language processing*, the grant H2020-ICT-2014-1-645442 (QT21) and Charles University Research Programme "Progres" Q18+Q48.

The authors would like to thank Mārcis Pinnis and Raivis Skadiņš for advice, comments and suggestions. Also, Pēteris Ņikiforovs for the base code of the web-based matrix visualization.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Helcl, Jindřich and Jindřich Libovický. Neural Monkey: An Open-source Tool for Sequence Learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17, 2017. ISSN 0032-6585. doi: 10.1515/pralin-2017-0001. URL `http://ufal.mff.cuni.cz/pbml/107/art-helcl-libovicky.pdf`.

Junczys-Dowmunt, Marcin, Tomasz Dwojak, and Hieu Hoang. Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA, 2016. URL `http://workshop2016.iwslt.org/downloads/IWSLT_2016_paper_4.pdf`.

Klejch, Ondřej, Eleftherios Avramidis, Aljoscha Burchardt, and Martin Popel. MT-ComparEval: Graphical evaluation interface for Machine Translation development. *The Prague Bulletin of Mathematical Linguistics*, 104(1):63–74, 2015.

Koehn, Philipp. *Statistical Machine Translation*. Cambridge University Press, 2009.

Madnani, Nitin. iBLEU: Interactively debugging and scoring statistical machine translation systems. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 213–214. IEEE, 2011.

[5]Tartu University Translator: `http://neurotolge.ee`

[6]OpenNMT: Open-Source Neural Machine Translation: `https://github.com/OpenNMT/OpenNMT`

[7]T2T: Tensor2Tensor Transformers: `https://github.com/tensorflow/tensor2tensor`

Och, Franz Josef and Hermann Ney. A Comparison of Alignment Models for Statistical Machine Translation. In *Proceedings of the 17th conference on Computational linguistics*, pages 1086–1090. Association for Computational Linguistics, 2000. ISBN 1-555-55555-1.

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*, pages 311–318. Association for Computational Linguistics, 2002.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. Nematus: a Toolkit for Neural Machine Translation. *EACL 2017*, page 65, 2017.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

Zeman, Daniel, Mark Fishel, Jan Berka, and Ondřej Bojar. Addicter: What Is Wrong with My Translations? *The Prague Bulletin of Mathematical Linguistics*, 96:79–88, 2011.

Address for correspondence:
Matīss Rikters
`matiss@lielakeda.lv`
Rainis blvd. 19, Riga, Latvia

PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 109 OCTOBER 2017 51–60

QE::GUI – A Graphical User Interface for Quality Estimation

Eleftherios Avramidis

German Research Center for Artificial Intelligence (DFKI), Language Technology Lab, Berlin, Germany

Abstract

Despite its wide applicability, Quality Estimation (QE) of Machine Translation (MT) poses a difficult entry barrier since there are no open source tools with a graphical user interface (GUI). Here we present a tool in this direction by connecting the back-end of the QE decision-making mechanism with a web-based GUI. The interface allows the user to post requests to the QE engine and get a visual response with the results. Additionally we provide pre-trained QE models for easier launching of the app. The tool is written in Python so that it can leverage the rich natural language processing capabilities of the popular dynamic programming language, which is at the same time supported by top web-server environments.

1. Introduction

Understanding the performance of Machine Translation (MT) is a necessary task for the development of MT systems and assessing the usability of their output. A wide variety of methods has given the possibility to analyze the results of the MT output and provide useful insights about the quality of the results. Already from the first years of MT, automatic evaluation metrics have provided scores representing the quality of the translation by scoring the MT output against the references (Papineni et al., 2002; Lavie and Agarwal, 2007). Recently, the efforts on MT evaluation have been expanded to the field of Quality Estimation (QE) which is capable of providing numerical qualitative judgments on the MT output without access to reference translations (Blatz et al., 2004; Specia et al., 2009). Being independent of reference translations, QE can be useful in real-life applications. Additionally, its inherent diverse linguistic features, combined with machine learning, can provide deeper and more specific qualitative indicators.

Corresponding author: eleftherios.avramidis@dfki.de

Cite as: Eleftherios Avramidis. QE::GUI – A Graphical User Interface for Quality Estimation. The Prague Bulletin of Mathematical Linguistics No. 109, 2017, pp. 51–60. doi: 10.1515/pralin-2017-0038.

Whereas most methods for the development and evaluation of MT, including QE, have been available through open-source tools, there has been little effort towards making them easily applicable and user-friendly. Being oriented to researchers, most software can only be executed in the commandline of Linux environments and the installation effort can be non-trivial. Therefore, the use of these tools is confined to a relatively small group of experienced developers, while the entry barrier for newcomers is considerable. Additionally, despite the many levels of qualitative analysis, most results are presented only on a high level, with limited options for visualization. With QE::GUI, the software described in this paper, we attempt to solve some of these issues. We focus on QE and we aim at providing a graphical user interface (GUI) on top of common QE software. The GUI is offered via a web-based framework that can be hosted in any web-server and can be operated with a common browser. It provides the possibility for users to upload their data, have them analyzed with QE models and have visualized statistics over the results.

It must be noted that instead of providing a full-flegded solution, here we mostly aim at setting the standards for further development by suggesting a particular well-structured framework. Therefore, the current functionality is limited to only a few cases of QE analyses, that should nevertheless be easily extended to cover more scenarios depending on the needs of the research and potential end-users. Despite its limited initial functionality, the software complies with several engineering principles that allow extensibility and easy addition of features and functionality. Additionally, the entire framework is built within a unified architecture, and the backbone is written in a single programming language (Python), including the web interface, the database layer and the computational back-end.[1]

This paper is accompanying the open-source code[2] in order to provide more details on the functionality of the software. After comparing our efforts with previous works (Chapter 2), we outline the basic usage and the capabilities of the software (Chapter 3). The architecture and the design is given in Chapter 4. Finally, ideas for further work and conclusions are outlined in Chapters 5 and 6 respectively.

2. Related Work

Several tools related to MT evaluation provide graphical interfaces in order to aid the understanding of the scored MT Quality. First, we shortly review some systems which provide graphical user interfaces related to reference-based automatic metrics.

The **Experiment Management System** (EMS; Koehn, 2010) is a tool for organizing the training and testing of MT models based on the popular statistical system

[1]The web interface contains non-Python elements such as HTML templates and CSS stylesheets, whereas embedded external tools also use other languages.

[2]Code: `https://github.com/lefterav/qegui`
Demo of basic interface: `http://blade-3.sb.dfki.de:8100/qe/` username: `admin` ; password: `pbml2017`

Moses (Koehn et al., 2007). For the models tested on particular test-sets, it offers a web-based interface that displays the BLEU score per test-set. The output for every system can be further examined by displaying additional sentence-level statistics. EMS is a pipeline that combines scripts in different programming languages and the information is communicated with intermediate files. The web interface is written in PHP (albeit with no database layer), whereas the back-end scripts are mostly written in Perl and Bash. **ComparEval** (Klejch et al., 2015) is a tool that allows uploading files with MT output and displays tables and graphs on the performance of the system. Multiple MT systems can be compared in parallel with bar charts, while the progress of consequent development versions of the same system can be visualized through a linear chart. The tool also offers pairwise comparison between system pairs and sentence-level analytics, including sentence-level scores, word-level error annotations and n-gram scoring.

The only tool that offers some GUI for QE is a particular version of QuEst named QuEst-Online, as presented by Shah et al. (2014). This version allows users to access the tool remotely from a web browser. They can upload their source and translated text, which are consequently analyzed to produce qualitative features. These features are used with a pre-built QE model in order to predict an estimated quality score for the translated sentence, including ranking of multiple translations. The entire system combines three modules in several programming languages: PHP for the web interface, Java for the feature generation and Python for the machine learning, as opposed to our tool, which organizes all steps with the same programming language. Finally, in contrast to QuEst-Online, our tool offers visualizations and graphs for the results, an administration interface with configuration options for the models, whereas the requests are stored in a database and organized in a hierarchical structure, allowing future retrieval and examination of previous evaluations.

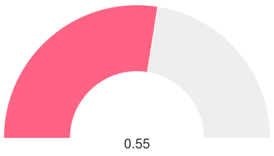
3. Usage

QE::GUI allows the users to submit documents translated by MT, including the source text and the MT output. These documents are analyzed and as a response, the user is given a set of statistics and graphs on the estimated quality of the translation.

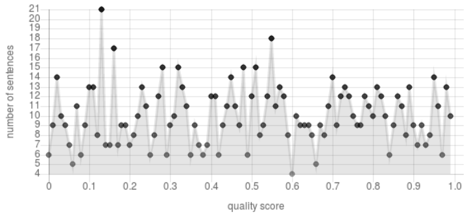
3.1. Organization of evaluation tasks

The interface is capable of storing the evaluation results in the long term, so that they can all be inspected in the future. Additionally, the user can apply hierarchical categorization to their uploads in order to maintain a clean structure of the contents.

The **document** is the basic organizational unit of the translations. Every document is a collection of source sentences with their respective MT outputs. The contents of a document can be uploaded from text files or inserted manually. All translated sentences in one document should be of the same language pair. The documents



(a) A gauge indicating the average score of all sentences in the document



(b) A linear area chart indicating mass distribution of the quality scores

Figure 1: Sample charts from the QE results

can be organized in evaluation tasks. Every **task** is effectively a folder which can contain many documents of different language pairs and sizes. The organization of documents into tasks is directly inflected on how the user can access the documents through the interface. After the list of the documents, the user is given the option to add an additional document.

3.2. Results and statistics

Once the document has been uploaded, a separate document page is created and the system starts the necessary operations for analyzing the sentences and performing the necessary calculations. When the calculations are finished, the document page gets populated with statistics about the estimated translation quality.

We show here two basic statistics accompanied with charts. For every evaluated document the page displays first the **average predicted score**, i.e. the mean score of all MT outputs. This is presented with a gauge which indicates the estimated score, as compared to the full range of the score. For instance, a predicted HTER score of 0.55 would be indicated by a gauge pointing a little more than half way in the range between 0 and 1 (Figure 1a). The indicator of the gauge can change color, depending on the level of quality.

Another possibly useful visual representation refers to the mass distribution of the sentence quality scores (Figure 1b). A linear chart indicates the amount of sentences that have been assigned to each possible quality score. This can be an indicator for how the quality of the translations ranges within the document. The conclusions can be complemented by observing a pie chart which indicated the distribution of the sentences scores among the 4 quartiles.

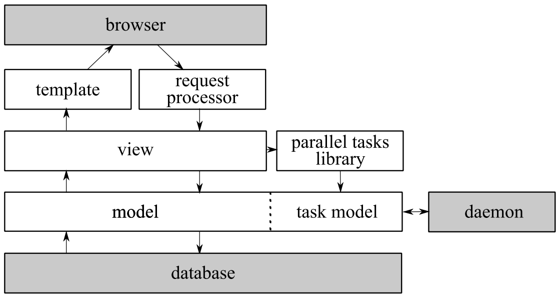


Figure 2: Generic internal architecture

3.3. Administration interface

The basic user interface has been deliberately designed so that it only contains the basic functionality for analyzing documents and observing the results of QE. Nevertheless, there is the possibility for advanced parametrization that falls out of the scope of simple usage. This is done through an advanced administration interface which contains functionality such as adding tasks, manually adding document sentences and translations, deleting tasks and documents and defining supported QE models and translation systems.

3.4. Deployment

The software can be deployed as a Python package and requires the creation of a Python virtual environment and the installation of some additional Python libraries. The QE framework may also require the installation of external natural language processing (NLP) required for the analysis of the text, prior to the application of machine learning. In its easiest form, the software can be run from a Django-compatible commandline bound on a SQLite database. For advanced scaling and better performance, it can be served as a proper website through common web servers (Apache, Nginx) with a more advanced database engine (MariaDB, PostgreSQL).[3]

4. Implementation

In this chapter we explain the main modules of the implementation. The generic architecture of the internal functionality is shown in Figure 2.

[3]Apache: `https://httpd.apache.org`, MariaDB: `https://mariadb.org`, Nginx: `https://nginx.org`, PostgreSQL: `https://www.postgresql.org`, SQLite: `https://www.sqlite.org`

4.1. Database structure

The implementation of the web-based interface and the underlying database structure is based on the latest version of the Django framework (ver. 1.11).[4] This has been chosen in order to take advantage of its powerful database modeling and its robustness, given the support of a wide developer community. Additionally, a wide range of libraries are easily available to extend the functionality. Last but not least, Django is fully Python-based, which makes it compatible and integrable with other Python applications. This is of a particular interest when it comes to QE, since several known open-source QE toolkits use Python (e.g. SciKit-Learn; Pedregosa et al., 2011) for their machine learning back-end.

As part of the Django framework, the web-based interface is built around a set of `models`, which are high-level representations of relational database tables. A simplified graphical representation of the models can be seen in Figure 3. To store the document structure presented above, we use the models `Task` and `Document`. A `Sentence` is the model which stores every source sentence and is associated with one or more MT outputs and reference translations through the models `MachineTranslation` and `ReferenceTranslation` (the latter is optional) respectively. Predicted quality scores for every MT output are stored in a `MachineTranslationEvaluation` model. The database also offers support for different evaluations of the same document (e.g. by different QE models),[5] which are organized by pointing from the separate `MachineTranslationEvaluation` instances to a `DocumentEvaluation` model.

4.2. Serving the data to the user

In order to allow access to the models, the framework employs a set of *v*iews (roughly a view for every page), which send queries to the models to retrieve the data and structure it as required for display. This data is then given to a set of templates, which take care of the setting text, objects, widgets and other elements in the final page shown to the user. To ease the representation of the page we use the responsive template framework Bootstrap, that reorders page elements depending on the screen size of the user's device. For creating the graphs, the data is passed to the open-source Javascript library JChart[6] which renders them in the HTML code of the page.

[4]Django :`http://djangoproject.com`

[5]The functionality of multiple evaluations is not available in the current preliminary version but is built into the database so that it can be extended in a next version.

[6]`https://django-jchart.matthisk.nl`

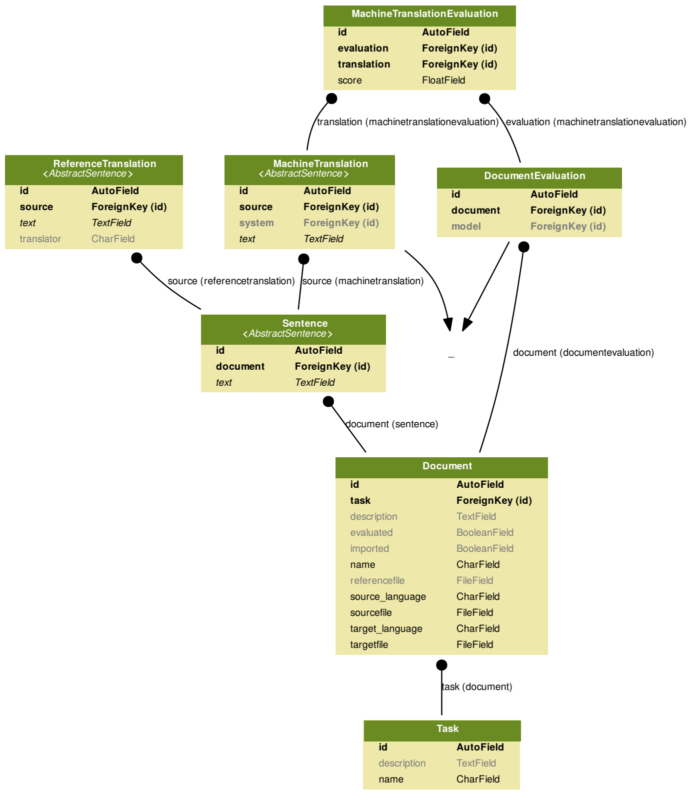


Figure 3: The structure of the Models that supports the web interface

4.3. Asynchronous data processing

A known issue for large NLP tasks is their scalability and low response times are a requirement for real-use applications. In our case the user is expected to upload a potentially large text file which contains source sentences and translations. The system is expected to store the sentences into the respective models of the database and then start processing them to estimate the quality scores. In order to avoid deadlocks and timeouts, we draw a line between the front-end (i.e. the interaction with the user) and the back-end (i.e. the processing of the data). The back-end tasks operate asynchronously, so that the loading of the front-end is not disturbed by time-consuming data-intensive processes. This way, the user can navigate away or close the document page but the processing they have requested will keep running in the background.

The implementation of the asynchronous processing uses the Django library `background tasks`. The steps followed are:

1. Every time the user uploads a new document, the library `background tasks` is instructed to create a new **asynchronous data processing task**.[7] The task is stored in a queue in the database, along with the parameters of the function and how it should be executed.
2. `Background tasks` provides a **background daemon**, which is then launched as a separated executable. The daemon regularly checks the queue and executes the pending tasks.
3. Having access to the Django database schema, the asynchronous tasks can deliver their results by populating the same models.
4. When every background task is finished, it notifies the front-end by enabling a boolean field in the relevant models.
5. The next time the front-end accesses the requested document evaluation, depending on the flag, will either proceed with visualizing the results, or display a message to the user to try again later.

It is also noteworthy that the daemon can run the queued tasks in parallel by launching several threads. The separate existence of the daemon also allows a distributed server set-up, where one server is responsible for the front-end and another one takes care of the back-end, but they both connect to the same database and use the same codebase.

4.4. Quality Estimation

In the last phase, a QE pipeline is triggered in order to process the data and deliver the estimation result. This pipeline consists of two major steps, that are accomplished by the existing QE toolkit: the feature generation and the application of a machine-learned model. During the feature generation, the quality estimation toolkit applies many NLP processes (e.g. language model scoring, parsing) in order to deliver numerical qualitative indicators (features). Then, these numbers are provided to a pre-trained QE model that delivers a *quality score*, i.e. a numerical judgment about the quality. The estimated quality score can then be stored in the model, associated with the respective MT output(s).

For the QE part, QE::GUI integrates existing code from other state-of-the-art Python-based tools. Qualitative (Avramidis, 2016) is used for the feature generation and prediction of sentence-level ranking, whereas models produced by the popular state-of-the-art QuEST (Shah et al., 2013) can be used for continuous score prediction. The database structure allows for loading and storing other Python-based QE models, provided that their usage is clearly documented.

[7] not to be confused with the *task* used for grouping documents in the part of the user interface

5. Further work

The presented software can cover functionality related to some basic use of QE and should still be considered as a preliminary version. Nevertheless, our aim is to suggest a unified framework that makes good use of state-of-the-art tools and is easily extensible to cover future needs. Further development can be directed towards covering more use-cases, including various types of QE (e.g. sentence binary filtering, word/phrase-level scoring). The graphical representation can be extended to compare the performance of different MT systems. Reference-based metrics can be integrated so that their scores can be displayed alongside the ones by QE, when reference translation are available. With the consideration of golden quality scores, this interface could also be used for automatically evaluating different QE approaches with correlation metrics. The principles of modularity and extensibility, along with the Git repository allow for wider collaboration and expansion of the development in a community scale.

6. Conclusion

We have introduced a new graphical user interface for Quality Estimation (QE) that exceeds any previous tool in terms of functionality and extensibility. It relies on a web-application built with Python Django. The user has the possibility to upload new documents to be analyzed by QE. The documents can be browsed through a user-friendly dashboard, categorized in tasks. For every document, the user can get the estimations by the QE along with several informative graphs and charts

The web interface is supported by a database layer which can store the data and its evaluation. Several state-of-the-art web libraries are seamlessly integrated to allow responsive appearance and drawing of charts. All data-intensive tasks, including QE, are executed asynchronously in a background queue by a separate daemon. Existing QE tools are integrated in order to perform feature generation and prediction of quality scores.

Acknowledgment This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement N° 645452 (QT21).

Bibliography

Avramidis, Eleftherios. Qualitative: Python Tool for MT Quality Estimation Supporting Server Mode and Hybrid MT. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 106:147–158, 2016.

Blatz, John, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 04)*, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Klejch, Ondřej, Eleftherios Avramidis, Aljoscha Burchardt, and Martin Popel. MT-ComparEval: Graphical evaluation interface for Machine Translation development. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 104:63–74, 2015. ISSN 0032-6585. doi: 10.1515/pralin-2015-0014. URL `https://ufal.mff.cuni.cz/pbml/104/art-klejch-et-al.pdf`.

Koehn, Philipp. An Experimental Management System. *The Prague Bulletin of Mathematical Linguistics*, 94:87–96, 2010. ISSN 1804-0462.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Chris Zens Richard a nd Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Lavie, Alon and Abhaya Agarwal. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, jun 2007. Association for Computational Linguistics.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, jul 2002. Association for Computational Linguistics.

Pedregosa, Fabian, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Shah, Kashif, Eleftherios Avramidis, Ergun Biçici, and Lucia Specia. QuEst: Design, Implementation and Extensions of a Framework for Machine Translation Quality Estimation. *The Prague Bulletin of Mathematical Linguistics*, 100:19–30, 2013.

Shah, Kashif, Marco Turchi, and Lucia Specia. An efficient and user-friendly tool for machine translation quality estimation. *LREC 2014. Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 3560–3564, August 2014.

Specia, Lucia, M. Turchi, N. Cancedda, M. Dymetman, and N. Cristianini. Estimating the Sentence-Level Quality of Machine Translation Systems. In *13th Annual Meeting of the European Association for Machine Translation*, pages 28–35, Barcelona, Spain., 2009.

Address for correspondence:
Eleftherios Avramidis
eleftherios.avramidis@dfki.de
German Research Center for Artificial Intelligence (DFKI GmbH)
Language Technology Lab
Alt Moabit 91c
10559, Berlin, Germany

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 109 OCTOBER 2017 61–91

CzeDLex – A Lexicon of Czech Discourse Connectives

Jiří Mírovský, Pavlína Synková, Magdaléna Rysová, Lucie Poláková

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

Abstract

CzeDLex is a new electronic lexicon of Czech discourse connectives, planned for publication by the end of this year. Its data format and structure are based on a study of similar existing resources, and adjusted to comply with the Czech syntactic tradition and specifics and with the Prague approach to the annotation of semantic discourse relations in text.

In the article, we first put the lexicon in context of related resources and discuss theoretical aspects of building the lexicon – we present arguments for our choice of the data structure and for selecting features of the lexicon entries, while special attention is paid to a consistent and (as far as possible) uniform encoding of both primary (such as in English *because*, *therefore*) and secondary connectives (e.g. *for this reason*, *this is the reason why*). The main principle adopted for nesting entries in the lexicon is – apart from the lexical form of the connective – a discourse-semantic type (sense) expressed by the given connective, which enables us to deal with a broad formal variability of connectives and is convenient for interlinking CzeDLex with lexicons in other languages.

Second, we introduce the chosen technical solution based on the Prague Markup Language, which allows for an efficient incorporation of the lexicon into the family of Prague treebanks – it can be directly opened and edited in the tree editor TrEd, processed from the command line in btred, interlinked with its source corpus and queried in the PML Tree Query engine.

Third, we describe the process of getting data for the lexicon by exploiting a large corpus manually annotated with discourse relations – the Prague Discourse Treebank 2.0: we elaborate on the automatic extraction part, post-extraction checks and manual addition of supplementary linguistic information.

Corresponding author: `mirovsky@ufal.mff.cuni.cz`
Cite as: Jiří Mírovský, Pavlína Synková, Magdaléna Rysová, Lucie Poláková. CzeDLex – A Lexicon of Czech Discourse Connectives. The Prague Bulletin of Mathematical Linguistics No. 109, 2017, pp. 61–91.
doi: 10.1515/pralin-2017-0039.

1. Introduction

In connection with rapid development of corpora annotated with discourse relations for different languages and in various frameworks (Carlson et al., 2003, Prasad et al., 2008 (English), Oza et al., 2009 (Hindi), Zeyrek et al., 2010, Zeyrek and Kurfalı, 2017 (Turkish), Al-Saif and Markert, 2010 (Arabic), Danlos et al., 2012 (French), Zhou and Xue, 2012, 2015 (Chinese), Stede and Neumann, 2014 (German), Iruskieta et al., 2013 (Basque), Da Cunha et al., 2011 (Spanish), to refer to just a few),[1] electronic lexicons of discourse connectives began to be built, although they are so far much less common. Electronic lexicons of discourse markers[2] are not only a useful tool in the theoretical research of text coherence/cohesion. Systematic information on discourse markers contributes to NLP tasks that involve processing of discourse relations (cf. e.g. Meyer et al., 2011, Stede, 2014 or Lin et al., 2014) and may help in machine translation, information extraction, text generation and other areas.

Our goal was to design and build an electronic lexicon of Czech discourse connectives, having in mind especially the following objectives:

- to contribute to the theoretical understanding of Czech connectives, and more generally, to understanding how text coherence/cohesion is established in Czech,
- to help in NLP tasks such as discourse processing, text generation and machine-translation, and
- to make the lexicon readable to a non-Czech speaker and linkable to existing lexicons of connectives in other languages.

There are several options how to actually build such a lexicon, i.e. how to fill it with data, from consulting existing printed lexicons, to using translation from lexicons in other languages, to exploiting existing discourse-annotated corpora in the given language. We have chosen the last option, as a large discourse-annotated treebank – the Prague Discourse Treebank 2.0 (Rysová et al., 2016) – is available for Czech.

The present article summarizes, updates and extends information on the design and build-up of the lexicon of Czech discourse connectives – CzeDLex – that was previously given in Mírovský et al. (2016b) and Synková et al. (2017, in print).

The subsequent text is organized as follows: Section 2 gives an overview of related research and existing lexicons of discourse connectives and compares main properties of CzeDLex and the other resources. In Section 3, the Prague Discourse Treebank 2.0 is introduced. Section 4 specifies basic terms such as "connective" and describes the lexicon structure from the theoretical point of view, providing reasons for decisions

[1] See also a list of discourse annotated corpora compiled within the COST TextLink project: `http://www.textlink.ii.metu.edu.tr/corpus-view`.

[2] We use "discourse markers" as a broader term for expressions structuring discourse, and "discourse connectives" (DCs) as a narrower term for expressions signalling semantico-pragmatic relations between two abstract objects (see 4.1).

behind the lexicon design. Section 5 describes the data format and application framework selected for the implementation of the lexicon, and presents also the process of extracting the lexicon from the data of the Prague Discourse Treebank 2.0, including subsequent checks, manual corrections and additions.

2. Related Research, Other Lexicons of Connectives

In this section, we put CzeDLex in context of current lexicography and compare it to other existing lexicons of connectives or expressions to a certain extent overlapping with some types of connectives.

Generally, lexicons (or dictionaries) may be of various kinds, reflecting different linguistic aspects. Traditionally, lexicons are characterized according to the number of languages they involve (monolingual, bilingual, multilingual dictionaries), their coverage (a general dictionary, a dialect dictionary, a sociolect dictionary reflecting e.g. colloquial language, adolescent language etc.), aspects of linguistic structure (an orthographic dictionary, a pronunciation dictionary, a frequency dictionary, a phraseological dictionary), the segment of the vocabulary (a dictionary of neologisms or a loan-word dictionary) or the group of users (a language learner's dictionary). For more details, see Hausmann (1985).

In this respect, lexicons of discourse markers/connectives represent a part of a specific lexicographic domain: in contrast to the majority of dictionaries/lexicons, they describe the synsemantic part of vocabulary (i.e. grammatical words, function words). As such, these lexicons are in fact lists of possible forms that can express one certain function in a language. These functional lexicons are so far much rarer and even more so in the Czech context. For other languages, there are similarly targeted lexicons, let us mention e.g. German lexicographic projects: Lexikon deutscher Konjuktionen (Buscha, 1989), Lexikon deutscher Partikeln (Helbig, 1988), Präpositionen (Schröder, 1986), Modalwörter (Helbig and Helbig, 1990) etc. Regarding the connective/discourse marker category, the printed resources include Dictionary of link words in English discourse (Ball, 1993), or the German two-volume Handbuch der deutschen Konnektoren (HdK, Pasch et al., 2003; Breindl et al., 2015).

Another specificity of the last years is the machine-readable form of such functional resources and the intention (and often the primary goal) to use these resources in various NLP tasks. Apart from the "standard" digitalized monolingual or translation dictionaries for a large scope of users, there are, mostly corpus-based, electronic projects assembling vocabulary with a specific function (e.g. evaluative language in the Czech SubLex, Veselovská and Bojar, 2013), or mining morphosyntactic annotation, e.g. valency properties of verbs (CzEngVallex, Urešová et al., 2016, for Czech and English) and similar.

CzeDLex may thus be described as an electronic corpus-based resource of Czech discourse connectives, containing English connective equivalents, reflecting written journalistic Czech language of the PDiT 2.0 texts (see Section 3) that provides functional descriptions of the expressions and phrases it covers. This includes morphosyntactic information, usage and meanings of the connectives in particular contexts and their frequencies in the underlying dataset (for more details, see Section 4.3).

Such a placement in the general typology of lexicographic projects puts CzeDLex right next to other newly emerging electronic lexicons of discourse markers or connectives. As far as we know, there are nowadays only few such projects, but the field is quickly growing and new projects arise every year now.[3] Perhaps one of the oldest electronic lexicons of discourse markers was the first version of DiMLex for German (Stede and Umbach, 1998), further, there is LexConn for French (Roze et al., 2012), DPDE for Spanish,[4] LICO for Italian (Feltracco et al., 2016) and others.[5] As some aspects of these lexicons served as a source of inspiration for the development of CzeDLex, we describe these lexicons and especially DiMLex in more detail later in this section.

From the Czech lexicographic projects, CzeDLex can be partly compared to the work of F. Čermák (2007, 2009). Secondary connectives in CzeDLex (i.e. expressions such as *z tohoto důvodu* [*for this reason*], for details see Section 4.1.1 below) to some extent overlap with phrases and idioms that are elaborated for Czech in his lexicon of Czech phrases and idioms. It consists of four volumes, dealing with 1. Comparisons, 2. Non-verbal expressions, 3. Verbal expressions and 4. Sentential expressions (see Čermák, 2009). Secondary connectives and Čermák's phrases and idioms presented in the lexicons overlap only slightly, but it is interesting to look at how these expressions are treated in various approaches.

The lexicon of phrases and idioms in Czech contains full and reduced lexicon entries. The full ones are for frequent expressions and the reduced ones for expressions with a lower frequency. The full entries contain various types of linguistic information such as stylistic characteristics, grammatical characteristics, intonation, context, valency, explanation of meaning, exemplification, synonyms or foreign language equivalents. The choice of entries (sorted in the alphabetical order) is based on corpus data (which is the same for CzeDLex). In this way, the lexicon aims to describe the current situation in the field of phraseology.

As an example of a sentential phrase in Čermák's lexicon, we find e.g. the phrase *Mám k tomu své/svý důvody.* [lit.: *I have my reasons for this.*] (which in PDiT 2.0 functions as a connective and was therefore included into CzeDLex under the connective phrases containing the word *důvod* [*reason*]). For a given phrase, the lexicon of phrases and idioms provides an explanation of its meaning, a context in which the phrase may

[3] To support building of such inventories of connectives in different European languages and to devise ways of interlinking their entries is one of the goals of the COST TextLink project, see `http://textlink.ii.metu.edu.tr`.

[4] `http://www.dpde.es`

[5] Compare also a list of inventories of discourse-structuring devices at `http://www.textlink.ii.metu.edu.tr/dsd-view`.

be used, and a synonymous construction. CzeDLex approaches similar phrases from a different perspective, namely in terms of coherence (i.e. we focus on the function the phrase has for the text coherence). Therefore, in CzeDLex, we deal with semantic discourse types expressed by the phrase, its Czech synonyms and English equivalent/s.

In the rest of this section, we compare the most important properties of some other existing connective lexicons to the properties of CzeDLex. During the design process of CzeDLex, the points of departure of similar projects were particularly important because of future lexicon interlinking and their usability for translation. We therefore aimed to be theoretically and technically as close to existing electronic lexicons of connectives as possible. As mentioned earlier, the main source of inspiration was the German machine-readable Lexicon of Discourse Markers, DiMLex (Stede and Umbach, 1998), developed in Potsdam and continuously enhanced (DiMLex 2, Scheffler and Stede, 2016). CzeDLex and DiMLex are indeed closely related in several basic aspects:

- they are both encoded in an XML-based format,
- the core of the delimitation of the category of discourse connectives/discourse markers is very similar,
- both cover part-of-speech, syntactic and semantic properties of the items they describe,
- semantic properties of the connectives are described via highly compatible frameworks – the sense taxonomy used in the Penn Discourse Treebank (Prasad et al., 2008) vs. its close Prague variant,
- both reflect ambiguity issues and record also non-connective usages.

On the other hand, different development processes of these inventories and different grammatical tradition (mostly in morphology) in discourse marker description resulted in several discrepancies between the two projects: Regarding the development process, DiMLex is being developed since 1998 and it is largely inspired by the extensive research project Handbuch der Deutschen Konnektoren (HdK; Pasch et al., 2003). CzeDLex is based upon the Prague Discourse Treebank 2.0, its annotation of discourse relations, syntactic analysis and part-of-speech tagging principles. The definition of a connective in DiMLex adopts five criteria from the HdK, M1-M5,[6] but drops the M2 criterion, as several (cca 25) prepositions, or, more precisely, adpositions (also postpositions, e.g. *–halber* and "Zirkumpositionen", e.g. *um ... Willen*), were considered discourse connectives and added to the lexicon. The CzeDLex connective definition is based on the Penn Discourse Treebank (PDTB) definition as a predicate of a binary

[6] (M1) X cannot be inflected. (M2) X does not assign case features to its syntactic environment. (M3) The meaning of X is a two-place relation. (M4) The arguments of the relation (the meaning of X) are propositional structures. (M5) The expressions of the arguments of the relation can be sentential structures (Scheffler and Stede, 2016).

relation opening positions for two text spans as its arguments and signalling a semantic or pragmatic relation between them (see 4.1 for details or compare Mírovský et al., 2016b). Prepositions are so far not included, but CzeDLex covers also some frequent secondary connectives (similar to the "AltLex" category in the PDTB approach). Some earlier work on more complex connective expressions with referential components in Czech can be found in Poláková et al. (2012) and mainly in Rysová and Rysová (2015), and for German, a pilot study of the anaphoric connective *demzufolge* [best translated as *accordingly, as a result, consequently*] is given in Stede and Grishina (2016). DiMLex now contains 275 German connectives of current use and the authors claim that the coverage is complete.

Nesting of lexicon entries in DiMLex follows the syntactic category of discourse markers. In this aspect, lemmas of connectives in CzeDLex are structured differently, according to discourse types (senses) they convey. The latter approach is also taken in the French LexConn (Roze et al., 2012), cf. e.g. several entries for the expression *alors*.

Semantic properties of the connectives are described via very similar frameworks: a variant of the PDTB sense taxonomy – PDTB 3.0 – for DiMLex versus Prague adjustments of the PDTB version 2.0 (see Table 1) for CzeDLex. In addition, DiMLex 2.0 was recently enriched by semantic relations according to more discourse frameworks, it lists all possible semantic/pragmatic characteristics of a given connective token also according to the frameworks of the Rhetorical Structure Theory (RST; Mann and Thompson, 1988b) and the Segmented Discourse Representation Theory (SDRT; Asher, 1993), and the grammar book of Helbig and Buscha (1984).

3. Prague Discourse Treebank 2.0

The Prague Discourse Treebank 2.0 (Rysová et al., 2016) is built upon the data of the Prague Dependency Treebank (Hajič et al., 2006; Bejček et al., 2013), which is a richly annotated corpus with manual multilayer annotation of approx. 50 thousand sentences of Czech journalistic texts from 1990's. The Prague Dependency Treebank contains morphological information on each token and two layers of syntactic annotation for each sentence (shallow and deep structure), both layers are represented by dependency trees. Besides, there is an annotation of information structure, pronominal and nominal coreference, bridging anaphora and multiword expressions. Annotation of discourse relations was carried out on top of deep-syntactic trees (on the so called tectogrammatical layer, see Example 1 and Figure 1) and covers relations expressed by a surface-present connective (for a definition of connective, see 4.1).

The set of discourse types (see the complete list in Table 1) is inspired by the Penn Discourse Treebank 2.0 sense hierarchy (Prasad et al., 2008) and the syntactico-semantic labels used for representation of compound sentences on the tectogrammatical layer.

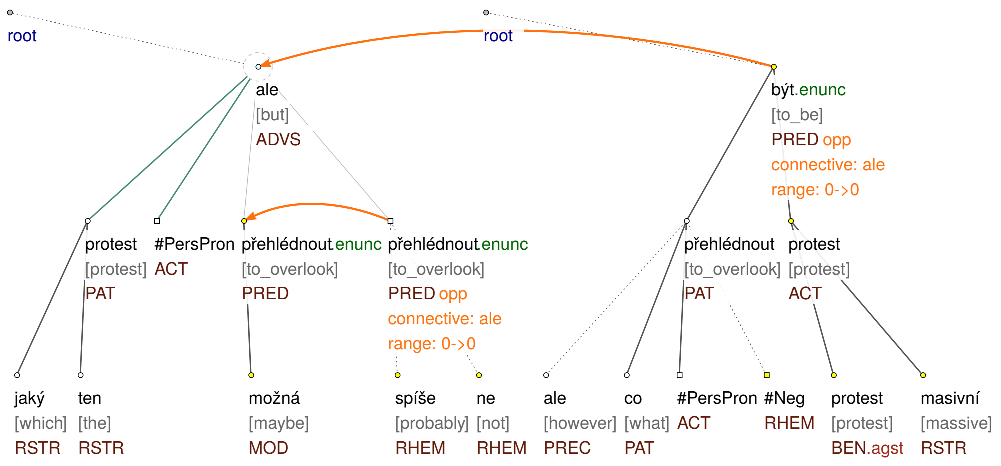


Figure 1. Example of an intra-sentential and an inter-sentential discourse relation in PDiT 2.0. Both relations are represented by thick curved arrows connecting roots of the arguments. Information about the semantic discourse types, connectives and range of the arguments is given at the starting nodes of the relations.

The first version of the annotation of discourse relations in the data of the Prague Dependency Treebank was published in 2012 as the Prague Discourse Treebank 1.0 (PDiT 1.0; Poláková et al., 2012b) and described in detail in Poláková et al. (2013).

(1) *Možná jsem nějaký ten protest přehlédl, **ale** spíše ne. Co jsem **ale** přehlédnout nemohl, byly masivní protesty proti protestům.* (PDiT 2.0)

[*Lit.: Maybe I overlooked some of the protests **but** probably not. What I **however** could not overlook, were massive protests against protests.*]

An updated version of the annotation of discourse relations of the same data was published in the Prague Dependency Treebank 3.0 (PDT 3.0; Bejček et al., 2013), with newly annotated second relations[7] and more systematic annotation of focusing particles (such as *also*, *too*) as parts of connectives of *conjunction* relation. A new attribute *discourse_special* was introduced to capture three special roles of phrases: headings

[7] Note that – unlike in the Penn Discourse Treebank approach – second relations annotated in the Prague Dependency Treebank 3.0 and in the Prague Discourse Treebank 2.0 only involve cases where different relations (in the term of semantic discourse type) between the same arguments are explicitly expressed by two different connectives (e.g. relations *opposition* and *asynchronous* expressed by connectives *but* and *then*, respectively, in the sentence *He wanted to go there **but then** he changed his mind.*). Second relations as they are understood in the Penn Discourse Treebank approach – i.e. two relations expressed by a single connective – are not annotated in our data.

CONTRAST	EXPANSION
confrontation	*conjunction*
opposition	*conjunctive alternative*
restrictive opposition	*disjunctive alternative*
pragmatic contrast	*instantiation*
concession	*specification*
correction	*equivalence*
gradation	*generalization*
CONTINGENCY	**TEMPORAL**
reason–result	*synchrony*
pragmatic reason–result	*precedence–succession*
explication	
condition	
pragmatic condition	
purpose	

Table 1. Semantic types of discourse relations in PDiT 2.0 and CzeDLex

(replaced the attribute *is_heading* from PDiT 1.0), metatext (text not belonging to the original newspaper text, produced during the creation of the corpus), and captions of pictures, graphs etc. (the updates were reported in Mírovský et al., 2014). Genres of documents were also annotated in the PDT 3.0 (and reported in Poláková et al., 2014). A detailed study dedicated to different aspects of discourse relations and coherence in Czech, elaborating on various types of annotations of discourse-related phenomena in the data of the Prague Dependency Treebank, can be found in Zikánová et al. (2015).

Annotations published in PDiT 1.0 and in the PDT 3.0 involved explicit discourse relations expressed by connectives belonging mostly to conjunctions, adverbs, particles and punctuation marks, some of them were formed also by multi-word phrases.[8] In 2014, discourse connectives were divided into primary and secondary according to their degree of grammaticalization (Rysová and Rysová, 2014, 2015), see 4.1.1 below.

[8] A detailed list of expressions involved in the PDiT 1.0 and PDT 3.0 annotations: (i) coordinating conjunctions: e.g. *a* [*and*], *ale* [*but*], *však* [*but*], (ii) subordinating conjunctions: e.g. *ačkoliv* [*although*], *protože* [*because*], (iii) particle expressions (including rhematizers): e.g. *ovšem* [*however*], *zkrátka* [*shortly*], (iv) adverbs: e.g. *potom* [*then*], *stejně* [*equally*], (v) some prepositions with demonstrative pronouns: e.g. *kromě toho* [*except for this*], *k tomu* [*in addition to this*], *tím* [*by this*], (vi) some types of idiomatic multiple-word connective means formed by linking of different expressions: e.g. *na jedné straně* [*on the one hand*], *stručně řečeno* [*in short*], *jinými slovy* [*in other words*], (vii) elements formed by letters or numbers expressing enumeration: e.g. a), b), 1., 2.; (viii) two punctuation marks: colon and dash (see Poláková et al., 2012a). These connectives are described in detail in Poláková (2015).

	PDiT 1.0 (2012)	PDT 3.0 (2013)	PDiT 2.0 (2016)
Primary connectives[9]	yes	updated	updated
Headings	yes	yes	yes
Second relations		yes	updated
Focusing particles		yes	yes
Captions, metatext		yes	yes
Genres of documents		yes	yes
Secondary connectives			yes

Table 2. Principal changes in the annotation of discourse relations and related phenomena in various published versions of the data. Each new version also brought fixes of annotation errors.

This new division is reflected in the newest published version of the Prague discourse annotation – the Prague Discourse Treebank 2.0 (PDiT 2.0; Rysová et al., 2016). Specifically, PDiT 2.0 contains a minor revision of the previous annotation (some types of connectives such as *kromě toho* [*except for this*] were re-annotated as secondary connectives) and annotation of discourse relations expressed by a new set of secondary connectives was added.

Table 2 summarizes the most significant changes of the annotation of discourse relations in the various versions of the published data. The last version – the Prague Discourse Treebank 2.0 – was used as the source data in the development of CzeDLex, as reported in the present article.

4. Theoretical Aspects

4.1. A Connective

One of the basic decisions in building a lexicon of discourse connectives concerns the delimitation of the connective category. In accordance with the Prague tradition of discourse annotation and the approach used for the annotation of PDiT 2.0, we understand a discourse connective as a predicate of a binary relation opening two positions for two text spans as its arguments and signalling a semantic or pragmatic relation between them.[10]

[9] We use the term "primary connectives" here in a simplified way, as this term was first used and defined in 2014. However, the annotations of explicit connectives in PDiT 1.0 (Poláková et al., 2012b) and the PDT 3.0 (Bejček et al., 2013) roughly correspond to this class of expressions; see footnote 8 for a detailed list.

[10] A similar approach was used in the PDTB, cf. Prasad et al. (2008).

The two connected text segments are defined according to Asher (1993) as abstract objects expressing events, states, situations, etc. Syntactically, abstract objects (discourse arguments) can be represented by various structures ranging from whole sentences or their combination, to simple clauses, to participial and infinitive constructions and nominal phrases. In PDiT 2.0, annotation of discourse relations was syntactically restricted to verbal arguments (i.e. whose basis is a finite verb).[11] CzeDLex therefore includes connectives in relations with verbal arguments only.

4.1.1. Primary and secondary connectives

Discourse connectives in PDiT 2.0 are divided into primary and secondary ones, according to Rysová and Rysová (2014), as already mentioned in Section 3. Primary connectives were defined as grammaticalized expressions such as *because* or *therefore* whereas secondary connectives were established as not (yet) fully grammaticalized structures with connecting function such as *except for this*, *the reason was* or *for this reason*.

CzeDLex contains both types of connectives. They, however, differ in many important aspects that need to be reflected in the lexicon design: lemmatization, syntactic characteristics, part-of-speech appurtenance, position of the arguments and argument integration (i.e. the position of a connective in the argument). Many secondary connectives may be inflected (*for this reason – for these reasons*; *the condition is – the conditions were* etc.) and they exhibit – at least in Czech – a high degree of variation (*důvod je* vs. *důvodem je* [*the reason is*: nominative vs. instrumental], both variants in Czech are equivalent).

4.1.2. Complex forms and modified connectives

Discourse connectives often occur in complex and/or modified forms (see Rysová, 2015). Complex forms consist of two or more connective words (i.e. words that can be connectives by themselves) that all participate on expressing the given discourse meaning (semantic discourse type, sense). Complex forms occur either in a single argument (*a proto* [*and therefore*]) or they may form correlative pairs (*buď nebo* [*either or*]).

Modified connectives contain an expression of an evaluative, modal or intensifying nature that further specifies/modifies the discourse relation, without changing its semantic type (*hlavně protože* [*mainly because*] or *možným důvodem je* [*a possible reason is*]).

[11] The annotation of secondary connectives in PDiT 2.0 took into account also arguments formed by noun phrases. These cases were annotated as notes at the core words of the secondary connectives, without the full annotation of the discourse relations (the whole connective, the arguments and their extent are not marked); these cases are not included in CzeDLex.

Both complex and modified connectives are included in CzeDLex, as parts of entries for the respective single connectives (for details and exceptions, see 4.3 below).

4.1.3. Non-connective usages

Most connective expressions (or, in case of secondary connectives, certain parts of them) exhibit a functional homonymy with expressions that have different functions in the text. Non-connective usages of these homonymous expressions can be categorized into several groups with specific properties:

- Expressions connecting mere entities (e.g. *towns **and** villages*) are not considered discourse connectives since they do not connect abstract objects (Asher, 1993).
- Expressions in the function of expressive, modifying or answer particles do not connect two abstract objects either, although their function belongs to the wider class of discourse markers in some contexts (e.g. ***So**, will you visit her? **Of course**.*).
- Homonyms of primary connectives sometimes function only as sentence constituents (mostly in the rhematic part of a sentence) and not as connectives (e.g. *Musíš to udělat úplně **jinak**.* [lit.: *You have to do it completely **otherwise**.*]).
  In contrast to the primary ones, the secondary connectives (or their parts) are always sentence constituents at the same time. However, their "core" words may also have a non-connective usage – cf. *The suggestion was rejected for procedural **reasons**.*

For each lexicon entry in CzeDLex, in addition to the list of connective usages, non-connective usages of the expression/phrase are listed at level two of the lexicon structure (see 4.2), along with their syntactic characteristics.[12]

4.2. Nesting of Lexicon Entries

The most important property of a discourse connective is its lexical form, and naturally the connectives in the lexicon are nested[13] on the first level (level one) according to their lemmas (which need to be representatively chosen for complex or modified connectives, and especially for secondary connectives, see below in 4.3).

Since we are building a lexicon of *discourse* connectives, the second most important property of a connective is the semantic discourse type the connective can convey (more precisely, a list of the discourse types). Therefore, on the second level (level two) of the lexicon structure, the entries are nested according to these semantic dis-

[12] A detailed analysis of "the degree of connectivity" of frequent Czech connectives according to the PDT 3.0 annotation can be found in Zikánová et al. (2015, pp. 161–162).

[13] By "nested" we mean organized, divided into individual entries.

course types.[14] This approach is justified also by a practical consideration: one of the primary uses of a lexicon of discourse connectives is machine translation. Interconnected lexicons of discourse connectives in different languages may help choose a correct translation of a connective in the given context (see e.g. Meyer and Poláková, 2013). The following observations suggest an answer to the question "which items (parts of records in the lexicons) should get connected?".

For translating a discourse connective to another language, it is not sufficient to only know the connective itself; for example, if we look up a translation of the English connective *while* into Czech in a publicly available online translation dictionary,[15] we get the following list:

- *zatímco*; *když* (synchronous events)
- *když*; *během toho, co* (synchronous events)
- *zatímco*; *kdežto*; *ale* (adversative relation [but])
- *i když*; *ačkoli*; *přestože* (concession [although])
- *(nějaký) čas*; *chvíle*; *chvilka* (noun)

...which is an ambiguous result and – as we can see – most of the options differ in the semantics of the connectives, which is very close to the discourse semantic type.[16]

On the other hand, to correctly translate a connective in context, it is not sufficient to know only the semantic discourse type the connective conveys either: if we try to "translate" the connective *while* based only on the fact that it is – in the given case – expressing e.g. the sense of *Contrast* in the PDTB taxonomy, and if we assume that in the Prague taxonomy the respective discourse type is *opposition*, we will find out that in PDiT 2.0, the relation of *opposition* is realized by 103 different connectives[17] – the most frequent of them are listed in Table 3.

We can conclude that to select a proper translation of a discourse connective, we need both the lexical information (the connective itself) and the semantic discourse type conveyed by the connective in the given context. This supports the chosen approach of nesting entries in the lexicon according to lexical forms of the connectives (level one) and semantic discourse types they convey (level two). These level-two entries are then to be mapped to their counterparts in other lexicons.[18]

---

[14] Non-connective usages of the connective words are nested according to their part of speech.

[15] `https://slovnik.seznam.cz`

[16] The part of speech of the connectives would not be of much help here – only one option ("noun") would be ruled out. For most other connectives, the part of speech would not help at all.

[17] including variants, complex forms and modifications

[18] There are of course many remaining issues. The linking is still not 1:1, lexicons use different definitions of "connectives", different taxonomies of semantic discourse types, different lists of features for entries in the lexicons, etc.

connective	count	connective	count
však	1 104	*nicméně*	36
ale	955	*sice ... však*	35
ovšem	197	*přitom*	32
sice ... ale	122	*aniž*	21
jenže	44	*a*	16
avšak	41	...	

Table 3. Most frequent connectives in PDiT 2.0 expressing the relation of opposition.

4.3. Connective Properties in CzeDLex

Based on the above considerations, the entries in CzeDLex are nested according to a two-level principle. We describe in detail properties of entries on these two levels here in 4.3.1 and 4.3.2.

4.3.1. Level-one

The level-one entry in the lexicon structure is represented by the lemma of the connective. Whereas selecting a representative lemma for primary connectives is usually a straightforward decision (see 4.3.2 for details about complex connectives), a suitable solution needs to be carefully thought of for secondary connectives.

There are, for example, many secondary connectives containing the word *reason* (*for this reason, that is the reason why, the reason is* etc.). We can consider the word *reason* their common "core" word, i.e. the word that most strongly signals the relation that the whole secondary connective expresses. In the lexicon structure, we group secondary connectives under lemmas of these "core" words, which are mainly nouns (*reason, condition, conclusion* etc.), secondary prepositions (*due to, because of, thanks to* etc.) and verbs (*to precede, to conclude, to sum up* etc.)

The first level entry as a whole is encoded in the element[19] ***lemma*** and contains the following information:

- element ***text***: the lemma of the connective
- element ***english***: an approximate English translation for a basic orientation; more precise translations are given in connection with semantic discourse types at level-two entries
- element ***type***: the type of the connective: *primary* vs. *secondary* (see 4.1.1)

[19] Some properties of the lexicon entries are encoded as XML elements, others as their attributes (see Section 5).

- element ***struct***: the structure of the connective: it signals whether the connective is *single* such as *proto* [*therefore*] or *complex* such as *jednak jednak* [*on the one hand on the other hand*]. The complex connectives are further differentiated in the attribute ***type***[20] according to their placement in the argument(s): complex connectives with parts occurring in both arguments (e.g. *jednak jednak* [*on the one hand on the other hand*] or *buď nebo* [*either or*]) are labeled *correlative*, while complex connectives with all parts occurring in a single argument are labeled *continuous* if no word can be inserted between the parts of the connective (e.g. the connective *i když* [*even if, although*]), or *discontinuous* if other words can occur between the connective parts (e.g. *a potom* [*and then*]).
- element ***variants***: a list of variants of the connective: they are further specified in the attribute ***type*** as *stylistic* (cf. neutral *tedy* [*so.neutral*] vs. informal *teda* [*so.informal*]) or *orthographic* (e.g. *mimoto* vs. *mimo to* [both meaning: *besides*]), or *inflection* (e.g. the form *čímž* [*by which*] is the instrumental form of the connective with the nominative form *což* [*which*])
- element ***conn-usages***: a list of connective usages – level-two entries
- element ***non-conn-usages***: a list of non-connective usages – level-two entries
- attribute ***id***: a lexicon-wide unique identifier of this level-one lexicon entry

4.3.2. Level two

For each level-one entry in the lexicon structure, its connective and non-connective usages are represented as level-two entries. In connective-usages, the discourse type is used as the base for nesting (reasons for this decision were given in 4.2), while in non-connective-usages (see 4.1.3), the part-of-speech appurtenance of the expressions is used.

If this rule were followed strictly, the depth of the lexicon structure for secondary connectives would increase to three levels, as these connectives often form different syntactic structures conveying the same discourse type that cannot be treated in a single unit – for example, both secondary connectives *for the following reason* and *that is the reason why* express the same semantic discourse type (*reason–result*) but differ in the argument semantics, i.e. the former signals the reason, while the latter signals the result (see the element *arg_semantics* below).

To keep the data structure identical both for primary and secondary connectives,[21] we keep the two-level structure also for the secondary connectives; they are therefore nested not only according to the discourse type they express, but also to their representative dependency scheme. This scheme is a general pattern for the connective structure – e.g. the secondary connectives *z tohoto důvodu* [*for this reason*], *z uvedených*

[20] It is an attribute *type* of the element *struct*, different from the element *type* above.

[21] which, for example, simplifies searching in the lexicon in the PML-Tree Query system (see Section 5)

důvodů [*for the given reasons*] or *z těchto důvodů* [*for those reasons*] are represented by the dependency scheme "z ((anaph. Atr) důvod.2)", i.e. a preposition *z* [*for*] plus an anaphoric attribute and the word *důvod* [*reason*] in genitive.

The second level entry of the lexicon is encoded in the element ***usage*** and contains the following information:

- element ***sense***: the discourse type (see possible values in Table 1)
- element ***scheme***: the dependency scheme (used for secondary connectives only)
- element ***gloss***: a Czech expression disambiguating the meaning of the connective (a synonym or an explanatory phrase)
- element ***english***: an English translation (the gloss in English)
- element ***pos***: the part-of-speech appurtenance of the connective (the lemma) in the given usage. Conjunctions are further distinguished in the attribute ***subpos*** as *coordinating* or *subordinating*.
- element ***syntax***: for secondary connectives, the part-of-speech characteristics of the core word is accompanied by a syntactic characteristics for the whole secondary connective represented by this usage (*nominal phrase*, *adjectival phrase*, *pronominal phrase*, *clause*, *adverbial phrase*, or *prepositional phrase*).
- element ***arg_semantics***: this characteristics specifies the semantics of the argument the connective occurs in. From the semantic perspective, there is a basic difference between symmetric and asymmetric discourse relations. While both arguments of a symmetric relation (i.e. *conjunction* or *synchrony*) share the same general semantic characteristics, asymmetric discourse relations (e.g. *reason–result* or *gradation*) hold between arguments that have different semantic nature (e.g. one argument expresses the reason, the other the result).[22] A connective of an asymmetric relation is characterized by its placement in one specific part of the relation it signals. For example, the coordinating conjunction *tedy* [*thus*] signals the result, while *totiž* [*because*] signals the reason. Similarly, the subordinating conjunctions *než* [*until*] and *když* [*when*] can be used for signalling *precedence–succession* – the former occurs in the argument expressing the event happening later, while the latter occurs in the argument expressing the earlier event. Table 4 gives an overview of all possible values for the attribute *arg_semantics*. For sym-

[22] In some approaches, the discourse types of the relations are different (e.g. Sanders et al. (1992) distinguish *Cause-Consequence* and *Consequence-Cause*, the PDTB 2.0 (Prasad et al., 2007) differentiates *Cause:reason* and *Cause:result* according to the argument order), in other approaches the relation remains the same, but some conventions marking ordering of the reason and the result are applied (e.g. in the Prague approach, there is only one *reason–result* relation, but the reason part of the relation is indicated by the starting point of the arrow (cf. Zikánová et al., 2015); the ISO standard (Prasad and Bunt, 2015) introduces only one *Cause* relation as well, the asymmetry of the relation is represented by specifying argument semantics in the definition of the relation). In the Rhetorical Structure Theory (Mann and Thompson, 1988a), the difference in the (a)symmetry of relations is captured by the feature of nuclearity (symmetric relations are multinuclear, while asymmetric ones have a nucleus and a satellite).

relation	argument semantics
concession	*concession:expectation* *concession:contra-expectation*
condition	*condition:condition* *condition:result of condition*
correction	*correction:claim* *correction:correction*
explication	*explication:claim* *explication:argument*
generalization	*generalization:more specific* *generalization:less specific*
gradation	*gradation:lower degree* *gradation:higher degree*
instantiation	*instantiation:general statement* *instantiation:example*
pragmatic condition	*pragmatic condition:pragmatic condition* *pragmatic condition:result of pragmatic condition*
pragmatic reason-result	*pragmatic reason-result:pragmatic reason* *pragmatic reason-result:pragmatic result*
precedence-succession	*precedence-succession:precedence* *precedence-succession:succession*
purpose	*purpose:action* *purpose:motivation*
reason-result	*reason-result:reason* *reason-result:result*
restrictive opposition	*restrictive opposition:general statement* *restrictive opposition:exception*
specification	*specification:less specific* *specification:more specific*
all other relations	*symmetric*

Table 4. Possible values of the argument semantics (attribute arg_semantics).

metric relations, the element *arg_semantics* has the value *symmetric*. For complex correlative connectives forming level-one entries, the value is given for the second part of the connective.

- element ***ordering***: signals the linear order of the argument the connective occurs in (relatively to the other – external – argument).[23] In the majority of cases, ordering is connected with the part-of-speech characteristics – coordinating conjunctions, adverbs and particles are placed in the second argument in the linear order, while subordinating conjunctions can be placed in either of the arguments. There are, however, exceptions – e.g. the particle *nejenže* [*not only that*] which occurs always in the first argument – that justify incorporation of this characteristics as a separate element into the lexicon. The element ordering has one of these five values: *1* for connectives occurring only in the first argument, *2* for connectives in the second argument, *1 or 2* for connectives in the first or second argument, *1 and 2* for complex correlative connectives and *N/A* for secondary connectives forming a separate syntactic unit (e.g. *Důvod je jednoduchý.* [*The reason is simple.*]) and therefore occurring entirely between the arguments.
- element ***integration***: captures the position of the connective within the argument. According to their origin and other possible functions in text, Czech connectives have different positions in the argument. Only subordinating conjunctions and prototypical coordinating conjunctions occupy the very beginning of the clause or sentence; the position of other connectives varies. Some of them are placed typically at the clitic, i.e. second position (e.g. *však* [*however*]), some of them are typically either on the first or on the second position (e.g. *potom* [*then*] or *proto* [*therefore*]) and for the class of focusing particles (i.e. expressions like *také* [*also*] or *jenom* [*only*]), the position is given by the information structure. For secondary connectives represented by the whole clause, *integration* is again *N/A*. Other values of this element, as follows from examples just mentioned, are *first*, *second*, *first or second*, and *any*. For complex correlative connectives forming level-one entries, the value is given for the second part of the connective only.
- element ***realizations***: a list of non-modified and non-complex secondary connectives from PDiT 2.0 represented by the given dependency scheme (applies only to secondary connectives)
- element ***modifications***: a list of the connective modifications: e.g. for the lemma *potom* [*then*] expressing *precedence–succession*, there is a modification *teprve potom* [*only then*]. Secondary connectives can be modified as well – cf. *hlavní důvod proč* [*the main reason why*]. Modifications are further distinguished in the attribute *type* as *eval* (evaluative), *modal*, and *intense* (intensifying).
- element ***complex_forms***: a list of complex connectives: e.g. for the lemma *potom* [*then*] expressing *precedence–succession*, there are for example complex forms *a potom* [*and then*] and *nejdřív potom* [*first then*]. Secondary connectives can have

[23] This differs from the original design reported in Mírovský et al. (2016b) where this element signalled the linear order of the external argument. The new semantics of this element is more consistent with the semantics of elements *arg_semantics* and *integration*.

complex forms as well – cf. *a z tohoto důvodu* [*and for this reason*]. The criterion for a complex form to be placed in the level-two entry under a certain lemma is the ability of the basic connective (the given lemma) to express the same discourse type. It means that e.g. the complex connective *přesto však* [*yet however*] expressing the discourse type of *concession* is placed in respective level-two entries under both lemmas *přesto* [*yet*] and *však* [*however*], because both these single connectives individually also express the discourse type of *concession* in PDiT 2.0. Further, according to its placement either in both arguments or in one argument, each complex form is labeled in the attribute *type* as *correlative*, *continuous* or *discontinuous* (see above among the level-one entry characteristics).

- element ***examples***: a list of a few illustrative examples from PDiT 2.0 and their English translations. Both intra-sentential and inter-sentential examples are – if available in the corpus – given for the connective usages and marked as such in the attribute ***type*** (*intra* vs. *inter*).
- element ***is_rare***: signals a rare use of the connective with the given discourse type
- element ***register***: captures whether the connective is used in the *neutral*, *formal* or *informal* register
- attribute ***id***: a unique identifier of this level-two entry

For non-connective usages, the argument semantics, ordering, integration, modifications and complex forms are not applicable, whereas other characteristics are given similarly as for connective usages.

4.3.3. Corpus frequencies

Numbers of occurrences in PDiT 2.0 were added to all individual variants, complex forms, modifications and realizations, as well as to connective and non-connective usages (level-two entries) and the whole lemmas (level-one entries), in two attributes: ***pdt_count*** and ***pdt_intra***, capturing numbers of all vs. intra-sentential occurrences of the respective items.

Contrary to our former intention (stated in Mírovský et al., 2016b) to extract the lexicon from 9/10 of the source corpus only (leaving the last 1/10 of the data for test purposes), we decided in the end to use the whole PDiT 2.0 for the extraction, to have the whole data of the corpus covered and interconnected with the lexicon.[24] All numbers in the attributes *pdt_count* and *pdt_intra* therefore reflect frequencies from the whole PDiT 2.0.

[24] Similarly to e.g. PDT-Vallex, a lexicon of valency frames of verbs and (newly) some nouns in the Prague Dependency Treebank (see Urešová, 2011 and Kolářová, 2014), which also covers the whole treebank.

5. Practical Implementation

This section describes the implementation of the lexicon in the Prague Markup Language framework (PML, see Section 5.1 just below) and advantages this choice brings. We show details of the data format on several examples, to demonstrate a relative ease of using the PML formalism and possibly encourage others to use it in their practical research. We also describe steps in the process of extracting the lexicon from the Prague Discourse Treebank 2.0 and mention a few post-processing steps needed to improve the quality of the final data, and connective properties that needed to be inserted into the lexicon manually.

5.1. Prague Markup Language

The data format used in the Prague Discourse Treebank 2.0 is called the Prague Markup Language (PML, Hana and Štěpánek, 2012).[25] It is a data format used for many other treebanks developed in Prague or abroad, such as the Prague Dependency Treebank since version 2.0, the Prague Czech-English Dependency Treebank (Hajič et al., 2012), the Slovene Dependency Treebank (Džeroski et al., 2006), the Croatian Dependency Treebank (Berović et al., 2012), Ancient Greek and Latin Dependency Treebanks (Bamman and Crane, 2011), as well as all treebanks in the HamleDT project (Zeman et al., 2015), and many others.

The PML is an abstract XML-based format designed for annotation of richly linguistically annotated corpora, and especially treebanks. It is independent of a particular annotation schema and can capture simple linear annotations as well as annotations with one or more richly structured interconnected annotation layers, dependency or constituency trees, including external lexicons.

The PML framework offers the following advantages:[26]

- The data can be browsed and edited in TrEd, a fully customizable tree editor (Pajas and Štěpánek, 2008). TrEd is written in Perl and can be easily customized to a desired purpose by extensions that are included in the system as modules.[27]
- The data can be processed using scripts written in btred – a command line version of TrEd.
- The data can be searched in the PML-TQ (Prague Markup Language–Tree Query, Pajas and Štěpánek, 2009), a powerful, yet user friendly, graphically oriented system for querying any data in the PML.

[25] `http://ufal.mff.cuni.cz/jazz/PML`

[26] The PML framework brings also low level tools for data validation (against a PML schema) and libraries to load and save data. And, of course, as the PML format is technically an XML, any general XML tool can be used for the data as well.

[27] Such a module was used also for the annotation of discourse relations in PDiT, see Mírovský et al. (2010).

Using the PML framework presupposes representing the data in the PML format. Encoding a particular treebank in the PML requires:

- defining a PML-schema for each annotation layer of the data – this includes definition of tree node types, relations between the nodes, attributes for individual node types, values of the attributes,
- defining a stylesheet for the data – the stylesheet gives a full control over the way the data are displayed in the tree editor TrEd,
- and, optionally, defining macros – Perl scripts for manipulation with the data from within TrEd or btred; macros are often created to simplify the most common tasks done by the annotators.

The following listing is a short example from the PML-schema for CzeDLex, i.e. from the definition of the format of the lexicon data in the PML, namely the definition of the format for level-one entries (the lemmas):

```
01 <type name="c-lemma.type">
02   <structure role="#NODE">
03     <member as_attribute="1" name="id" role="#ID" required="1">
         <cdata format="ID"/></member>
04     <member as_attribute="1" name="pdt_count">
         <cdata format="nonNegativeInteger"/></member>
05     <member name="text" required="1"><cdata format="any"/></member>
06     <member name="english"><cdata format="any"/></member>
07     <member name="type" type="c-type.type"/>
08     <member name="struct" type="c-struct.type"/>
09     <member name="variants" type="c-variants.type"/>
10     <member name="usages" type="c-usages-all.type" role="#CHILDNODES"/>
11   </structure>
12 </type>
```

Notice the declarations of roles (`role="#NODE"`, `role="#CHILDNODES"`, lines 2 and 10), defining which data structures should be understood (i.e. represented) as tree nodes, and also the declaration of the identifier role (`role="#ID"`, line 3), defining which element should be understood as the key for the records.

Similar type definitions need to be provided for all other parts of the lexicon data structure, i.e. for the types referred to in the definition of the type `c-lemma.type` above and for all other data types needed in the lexicon. For example, the definition of the type `c-type.type` referred to from line 7 looks like this:

```
<type name="c-type.type">
  <choice>
    <value>primary</value>
    <value>secondary</value>
  </choice>
</type>
```

The following commented example shows the respective part of the resulting lexicon entry for the connective *potom* [*then, afterwards*]:

```
<lemma id="l-potom" pdt_count="95"> (a level-one entry)
  <text>potom</text>  (the lemma itself)
  <english>then; afterwards</english> (an approximate English translation;
    more precise translations are given at level-two entries)
  <type>primary</type>  (vs. secondary)
  <struct>single</struct>  (vs. complex)
  <variants>
    (no variants in the data for this lemma)
  </variants>
  <usages>
    <conn-usages pdt_count="80" pdt_intra="37">
      (list of connective usages, see Figure 2)
    </conn-usages>
    <non-conn-usages pdt_count="15">
      (list of non-connective usages)
    </non-conn-usages>
  </usages>
</lemma>
```

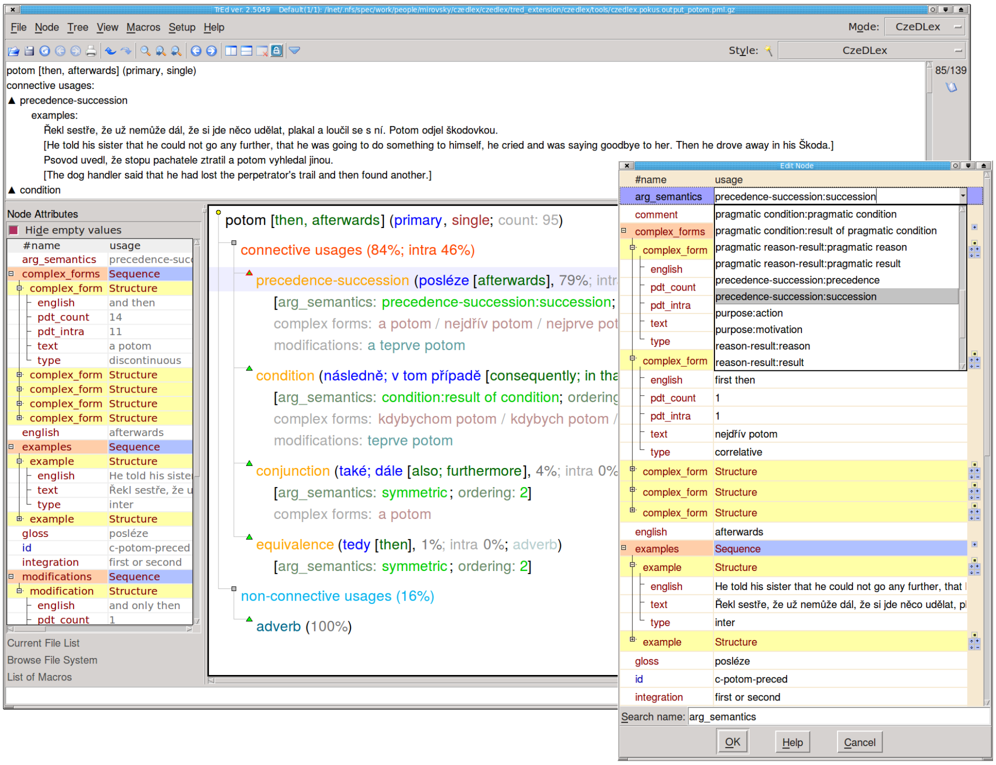
The commented example in Figure 2 shows a level-two entry in the PML for the lemma *potom* [*then, afterwards*], defining the lemma's connective usage with the semantic discourse type *precedence–succession*. The same part of the lexicon data is displayed in Figure 3 – it shows the lexicon loaded in the tree editor TrEd, allowing a user to inspect the record(s) or an annotator to make manual changes in the data. It displays the entry for the whole lemma, with an opened dialog window for editing the connective usage representing the discourse type *precedence–succession*, and a roll-down list of available options for the value of the element *arg_semantics*. The lemma (level-one entry), the list of connective usages, the list of non-connective usages, and the individual usages (level-two entries) are represented by tree nodes.

Using the PML for the lexicon brings, apart from the three advantages named earlier in this section, another possibility – the lexicon can be easily interlinked with the source data, i.e. the Prague Discourse Treebank 2.0, by adding identifiers of the lexicon entries (values of the attribute *id*, e.g. *c-potom-preced* from the example in Figure 2, line 1) to the respective places in the treebank, using so called PML references. The query system PML-TQ then allows for incorporating information both from the treebank and the lexicon into a single query, allowing – for example – to search for:[28]

[28] See Mírovský et al. (2014) and Mírovský et al. (2016a) for examples of using the PML-TQ for searching in discourse-annotated treebanks (the PDT 3.0 and the PDTB 2.0, respectively).

```
<usage id="c-potom-preced" pdt_count="63" pdt_intra="30">
  <sense>precedence-succession</sense>  (the represented semantic discourse type)
  <gloss>posléze</gloss>  (a synonym/explanation of the meaning in Czech)
  <english>afterwards</english>  (English translation)
  <pos>adverb</pos>  (part of speech)
  <arg_semantics>precedence-succession:succession</arg_semantics>
    (the argument associated with the connective represents
     the ``subsequent'' part of the relation)
  <ordering>2</ordering>  (the argument associated with the connective is placed
                           second in the surface order of the arguments)
  <integration>first or second</integration>  (a typical position in the argument)
  <register>neutral</register>  (vs. formal, informal)
  <modifications>  (a list of modifications)
    <modification type="intense" pdt_count="1" pdt_intra="1">
      <text>a teprve potom</text>  (an intensifying modification)
      <english>and only then</english>
    </modification>
  </modifications>
  <complex_forms>  (a list of complex forms)
    <complex_form type="discontinuous" pdt_count="14" pdt_intra="11">
      <text>a potom</text>
      <english>and then</english>
    </complex_form>
    (four more complex forms omitted to save space here)
  </complex_forms>
  <examples>  (a list of examples from PDiT 2.0)
    <example type="inter">  (an inter-sentential example)
      <text>Řekl sestře, že už nemůže dál, že si jde něco udělat, plakal
            a loučil se s ní. Potom odjel škodovkou.</text>
      <english>He told his sister that he could not go any further, that
            he was going to do something to himself, he cried and was saying
            goodbye to her. Then he drove away in his Škoda.</english>
    </example>
    <example type="intra">  (an intra-sentential example)
      <text>Psovod uvedl, že stopu pachatele ztratil a potom vyhledal jinou.</text>
      <english>The dog handler said that he had lost the perpetrator's trail
               and then found another.</english>
    </example>
  </examples>
  <pdt>  (information closely related to the source corpus)
    <discourse_type>preced</discourse_type>
    <pos_list>
      <pos>adverb</pos>
    </pos_list>
  </pdt>
</usage>
```

Figure 2. An abbreviated level-two entry for the lemma potom [then, afterwards] and the semantic discourse type precedence–succession.


TrEd ver. 2.5049 Default(1/1): /lnet/.nfs/spec/work/people/mirovsky/czedlex/czedlex/tred_extension/czedlex/tools/czedlex.pokus.output_potom.pml.gz
File Node Tree View Macros Setup Help
Mode: CzeDLex
Style: CzeDLex
85/139
Current File List
Browse File System
List of Macros
Edit Node
OK
Help
Cancel

potom [then, afterwards] (primary, single)
connective usages:
▲ precedence-succession
examples:
Řekl sestře, že už nemůže dál, že si jde něco udělat, plakal a loučil se s ní. Potom odjel škodovkou.
[He told his sister that he could not go any further, that he was going to do something to himself, he cried and was saying goodbye to her. Then he drove away in his Škoda.]
Psovod uvedl, že stopu pachatele ztratil a potom vyhledal jinou.
[The dog handler said that he had lost the perpetrator's trail and then found another.]
▲ condition

Node Attributes
Hide empty values

#name	usage
arg_semantics	precedence-succ
complex_forms	Sequence
complex_form	Structure
english	and then
pdt_count	14
pdt_intra	11
text	a potom
type	discontinuous
complex_form	Structure
complex_form	Structure
complex_form	Structure
complex_form	Structure
english	afterwards
examples	Sequence
example	Structure
english	He told his siste
text	Řekl sestře, že u
type	inter
example	Structure
gloss	posléze
id	c-potom-preced
integration	first or second
modifications	Sequence
modification	Structure
english	and only then
pdt_count	1

potom [then, afterwards] (primary, single; count: 95)
connective usages (84%; intra 46%)
precedence-succession (posléze [afterwards], 79%; intr
[arg_semantics: precedence-succession:succession;
complex forms: a potom / nejdřív potom / nejprve pot
modifications: a teprve potom
condition (následně; v tom případě [consequently; in tha
[arg_semantics: condition:result of condition; ordering
complex forms: kdybychom potom / kdybych potom /
modifications: teprve potom
conjunction (také; dále [also; furthermore], 4%; intra 0%
[arg_semantics: symmetric; ordering: 2]
complex forms: a potom
equivalence (tedy [then], 1%; intra 0%; adverb)
[arg_semantics: symmetric; ordering: 2]
non-connective usages (16%)
adverb (100%)

#name	usage
arg_semantics	precedence-succession:succession
comment	pragmatic condition:pragmatic condition
complex_forms	pragmatic condition:result of pragmatic condition
complex_form	pragmatic reason-result:pragmatic reason
english	pragmatic reason-result:pragmatic result
pdt_count	precedence-succession:precedence
pdt_intra	precedence-succession:succession
text	purpose:action
type	purpose:motivation
	reason-result:reason
complex_form	reason-result:result
english	first then
pdt_count	1
pdt_intra	1
text	nejdřív potom
type	correlative
complex_form	Structure
complex_form	Structure
complex_form	Structure
english	afterwards
examples	Sequence
example	Structure
english	He told his sister that he could not go any further, that
text	Řekl sestře, že už nemůže dál, že si jde něco udělat, pl
type	inter
example	Structure
gloss	posléze
id	c-potom-preced
integration	first or second

Search name: arg_semantics

Figure 3. CzeDLex opened in the tree editor TrEd, with the lemma potom [then, afterwards] displayed. In the left panel, as well as in the pop-up window on the right side, information for the selected connective usage with the semantic discourse type precedence–succession is available. In the pop-up window, a pull-down menu for a selection of the argument semantics is being used.

- all occurrences of discourse relations in the treebank expressed by connectives that have the ability to express (in different contexts) more than X (e.g. 2) different discourse types (senses),
- all occurrences of discourse relations in the treebank expressed by connective words that are ambiguous in their connective vs. non-connective usages,
- all occurrences of discourse relations in the treebank expressed by complex or modified connectives.

5.2. Data Extraction

The process of extracting a raw base of the lexicon from the Prague Discourse Treebank 2.0 started with an extraction of a list of all connectives annotated in the treebank data, using a simple PML-TQ query. In this all-connective list, each different string of words (e.g. *ale* [*but*] vs. *ale zároveň* [*but at the same time*] vs. *ale také* [*but also*]) formed a separate item. Primary and secondary connectives were already distinguished in the source corpus data and were treated separately. In over 20 thousand annotated discourse relations in the treebank, there were approx. 700 different items for the primary connectives and 350 for the secondary ones. Human annotators then manually divided the connectives into groups of connectives belonging to the same lemma, and in each group further distinguished complex forms, variants, modifications and (for the secondary connectives) realizations. For selected secondary connectives, also dependency schemes representing syntactically different realizations were created and the connectives were divided into subgroups according to the schemes.

This manually processed list served as an input for a btred[29] script that went through the whole data of the treebank, found all occurrences of the lemmas (and their variants, modifications etc.) and sorted them into the lexicon according to their type of usage (connective vs. non-connective) and the semantic discourse type of the relations (or the part of speech for non-connective usages). For each usage, a number of the shortest intra-sentential and inter-sentential examples[30] were collected (the annotators later chose the most suitable ones and added their English translations). Several other attributes could be set automatically as well – the part of speech, in most cases also the argument semantics and ordering (according to the orientation of the discourse arrow and position of the connective in an argument). Numbers of occurrences in PDiT 2.0 were added to all individual variants, complex forms and modifications, to connective and non-connective usages (level-two entries) and the whole lemmas (level-one entries).

After the lexicon was extracted from the annotated treebank, a few automatic or semi-automatic post-processing and data validity checking steps were performed. All counts of appearances of various lexicon data structures in the source treebank data were checked (e.g. if counts of individual connectives sum up to counts of the usages and the lemmas). Another important verifying step checked for each complex form (e.g. *ale také* [*but also*]) that its basic lemma (the respective level-one entry, say *ale* [*but*]) appeared in the treebank with the same discourse type. If not, the complex form was removed from that lemma (being for the moment left as a complex form of the other lemma forming the complex form, in our case *také* [*also*]). If the complex form was by

[29] a command line version of the tree editor TrEd

[30] For some connectives, only one type of examples could be found. The distinction also does not apply to non-connective usages.

this process removed from all its basic lemmas, a new level-one entry for this complex form was created, with the value *complex* in the element *struct*.

Several properties required manual work, as the treebank data either did not contain this information at all (English translations, Czech glosses, register, rareness, syntactic characteristics of secondary connectives) or the data were not big enough to cover all existing possibilities (dependency scheme, integration, sometimes ordering).

6. Conclusion

We have presented theoretical and implementation aspects of the design and development process of CzeDLex – a new electronic Lexicon of Czech Discourse Connectives. It is the first lexicon of Czech connectives and its uniqueness in the worldwide sense also lies in the fact that it includes not only primary but also secondary connectives. Special effort was dedicated to having both types of connectives represented in a relatively uniform way, as much as their different syntactic nature allows. We have also presented the data format used – the Prague Markup Language – and advantages this choice brings, and elaborated on the actual process of exploiting the source corpus, namely the Prague Discourse Treebank 2.0, to build the raw basis of the lexicon, with subsequent automatic and manual checks, corrections and additions.

Building the lexicon on the basis of an annotated corpus brings a certainty that the selection of the connectives and their coverage in the lexicon are to a certain degree representative but at the same time it sets limits on both these aspects, as the source treebank consists of newspaper texts only and, although it is large for a manually annotated treebank, its size is still limited.[31]

CzeDLex is built not only for theoretical purposes. Given its rich annotation of the properties of the connectives (including syntactic characteristics of the connectives, a general dependency scheme for the secondary connectives and distinction of variants, complex connectives and modified connectives), it may be useful also for NLP tasks that involve discourse parsing, for machine translation, information extraction and for text generation.

Our aim was also to make the lexicon readable for non-Czech speakers and to simplify its future interlinking with lexicons in other languages. We tried to achieve these goals by structuring the lexicon entries by semantic discourse types, by providing comprehensive morphological, syntactic and other characteristics both for the primary and secondary connectives, by using both human and computer readable format and by having all names of elements, attributes and their values (with the obvious exception of Czech word entries and Czech corpus examples) in English. In

[31] And much smaller than e.g. the SYN series of the Czech National Corpus, which contains automatically morphologically annotated texts in size of approx. 100 million words.

addition, each entry in Czech was supplemented by its English translation, including all corpus examples.

The first version of CzeDLex will be published this year in the Lindat/Clarin repository[32] under the Creative Commons license. It will cover an essential part of the connectives used in the Prague Discourse Treebank 2.0.[33] The second version of CzeDLex, planned to be published next year, will cover all connectives annotated in the treebank.

Acknowledgements

The authors gratefully acknowledge support from the Ministry of Education, Youth and Sports of the Czech Republic (project COST-cz LD15052), and the Grant Agency of the Czech Republic (project GA17-06123S). The work was supported from European Regional Development Fund-Project "LINDAT/CLARIN" (No. CZ.02.1.01/0.0/0.0/16_013/0001781). The research reported in the present contribution has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071). The authors are also grateful for inspiration coming from meetings and work realized within the European project TextLink (COST Action IS1312).

Bibliography

Al-Saif, Amal and Katja Markert. The Leeds Arabic Discourse Treebank: Annotating Discourse Connectives for Arabic. In *Proceedings of LREC 2010*, pages 2046–2053, Valletta, Malta, 2010.

Asher, Nicholas. *Reference to abstract objects in discourse*. Kluwer, Norwell, MA, 1993.

Ball, Wilson James. *Dictionary of link words in English discourse*. Macmillan, 1993.

Bamman, David and Gregory Crane. The ancient Greek and Latin dependency treebanks. In *Language technology for cultural heritage*, pages 79–98. Springer, 2011.

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. Data/software, 2013.

Berović, Daša, Željko Agić, and Marko Tadić. Croatian dependency treebank: Recent development and initial experiments. In *Seventh International Conference on Language Resources and Evaluation (LREC 2012)*, 2012.

Breindl, Eva, Anna Volodina, and Ulrich Hermann Waßner. *Handbuch der deutschen Konnektoren 2: Semantik der deutschen Satzverknüpfer*, volume 13. Walter de Gruyter GmbH & Co KG, 2015.

[32] `http://lindat.cz`

[33] All those that will have undergone all checks and manual additions by that time.

Buscha, Joachim. *Lexikon deutscher Konjunktionen*. Langenscheidt, Verlag Enzyklopädie, 1989.

Carlson, Lynn, Daniel Marcu, and Mary Ellen Okurowski. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer, 2003.

Čermák, František. *Frazeologie a idiomatika: česká a obecná*. Karolinum, 2007.

Čermák, František. *Slovník české frazeologie a idiomatiky*. Leda, 2009.

Da Cunha, Iria, Juan-Manuel Torres-Moreno, and Gerardo Sierra. On the development of the RST Spanish Treebank. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 1–10. Association for Computational Linguistics, 2011.

Danlos, Laurence, Diégo Antolinos-Basso, Chloé Braud, and Charlotte Roze. Vers le FDTB: French Discourse Tree Bank. In *TALN 2012: 19ème conférence sur le Traitement Automatique des Langues Naturelles*, pages 471–478, 2012.

Džeroski, Sašo, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*, 2006.

Feltracco, Anna, Elisabetta Jezek, Bernardo Magnini, and Manfred Stede. LICO: A Lexicon of Italian Connectives. *CLiC it*, page 141, 2016.

Hajič, Jan, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, Magda Ševčíková-Razímová, and Zdeňka Urešová. Prague Dependency Treebank 2.0. Data/software, 2006.

Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, 2012. ELRA, European Language Resources Association.

Hana, Jirka and Jan Štěpánek. Prague Markup Language Framework. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 12–21, Stroudsburg, 2012. Association for Computational Linguistics, Association for Computational Linguistics.

Hausmann, Franz Josef. Lexikographie. *Handbuch der Lexikologie. Königstein: Athenäum*, pages 367–411, 1985.

Helbig, Gerhard. *Lexikon deutscher Partikeln*. Verlag Enzyklopädie, 1988.

Helbig, Gerhard and Joachim Buscha. *Deutsche Grammatik*. Verlag Enzyklopädie, 1984.

Helbig, Gerhard and Agnes Helbig. *Lexikon deutscher Modalwörter*. Verlag Enzyklopädie, 1990.

Iruskieta, M., M. Aranzabe, A. Diaz de Ilarraza, I. Gonzalez, I. Lersundi, and O. Lopez de Lacalle. The RST Basque TreeBank: an online search interface to check rhetorical relations. In *4th Workshop RST and Discourse Studies*, pages 40–49, Sociedad Brasileira de Computacao, Fortaleza, CE, Brasil, 2013.

Kolářová, Veronika. Valence vybraných typů deverbativních substantiv ve valenčním slovníku PDT-Vallex. Technical Report TR-2014-56, ÚFAL MFF UK, 2014.

Lin, Ziheng, Hwee Tou Ng, and Min-Yen Kan. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184, 2014.

Mann, William C. and Sandra A. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8:243–281, 1988a.

Mann, William C. and Sandra A. Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281, 1988b.

Meyer, Thomas and Lucie Poláková. Machine translation with many manually labeled discourse connectives. In *Proceedings of the 1st DiscoMT Workshop at ACL 2013 (51st Annual Meeting of the Association for Computational Linguistics)*, pages 43–50, Sofia, Bulgaria, 2013.

Meyer, Thomas, Andrei Popescu-Belis, Sandrine Zufferey, and Bruno Cartoni. Multilingual annotation and disambiguation of discourse connectives for machine translation. In *Proceedings of the SIGDIAL 2011 Conference*, pages 194–203. Association for Computational Linguistics, 2011.

Mírovský, Jiří, Lucie Mladová, and Zdeněk Žabokrtský. Annotation Tool for Discourse in PDT. In Huang, Chu-Ren and Dan Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, volume 1, pages 9–12, Beijing, China, 2010. Chinese Information Processing Society of China, Tsinghua University Press.

Mírovský, Jiří, Pavlína Jínová, and Lucie Poláková. Discourse Relations in the Prague Dependency Treebank 3.0. In Tounsi, Lamia and Rafal Rak, editors, *The 25th International Conference on Computational Linguistics (Coling 2014), Proceedings of the Conference System Demonstrations*, pages 34–38, Dublin, Ireland, 2014. Dublin City University (DCU), Dublin City University (DCU).

Mírovský, Jiří, Lucie Poláková, and Jan Štěpánek. Searching in the Penn Discourse Treebank Using the PML-Tree Query. 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 1762–1769, Paris, France, 2016a. European Language Resources Association.

Mírovský, Jiří, Pavlína Synková, Magdaléna Rysová, and Lucie Poláková. Designing CzeDLex – A Lexicon of Czech Discourse Connectives. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation*, pages 449–457, Seoul, Korea, 2016b. Kyung Hee University, Kyung Hee University.

Oza, Umangi, Rashmi Prasad, Sudheer Kolachina, Dipti Misra Sharma, and Aravind Joshi. The Hindi Discourse Relation Bank. In *Proceedings of the third Linguistic Annotation Workshop*, pages 158–161, 2009.

Pajas, Petr and Jan Štěpánek. Recent Advances in a Feature-Rich Framework for Treebank Annotation. In Scott, Donia and Hans Uszkoreit, editors, *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 673–680, Manchester, 2008. The Coling 2008 Organizing Committee.

Pajas, Petr and Jan Štěpánek. System for Querying Syntactically Annotated Corpora. In Lee, Gary and Sabine Schulte im Walde, editors, *Proceedings of the ACL–IJCNLP 2009 Software Demonstrations*, pages 33–36, Suntec, 2009. Association for Computational Linguistics.

Pasch, Renate, Ursula Braußе, Eva Breindl, and Ulrich Hermann Waßner. *Handbuch der deutschen Konnektoren. Linguistische Grundlagen der Beschreibung und syntaktische Merkmale der deutschen Satzverknüpfer (Konjunktionen, Satzadverbien und Partikeln)*. Walter de Gruyter, 2003.

Poláková, Lucie. *Discourse Relations in Czech*. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 2015.

Poláková, Lucie, Pavlína Jínová, and Jirí Mírovskỳ. Interplay of Coreference and Discourse Relations: Discourse Connectives with a Referential Component. In *LREC*, pages 146–153. Citeseer, 2012.

Poláková, Lucie, Pavlína Jínová, Šárka Zikánová, Zuzanna Bedřichová, Jiří Mírovský, Magdaléna Rysová, Jana Zdeňková, Veronika Pavlíková, and Eva Hajičová. Manual for Annotation of Discourse Relations in Prague Dependency Treebank. Technical Report 47, Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic, 2012a.

Poláková, Lucie, Pavlína Jínová, Šárka Zikánová, Eva Hajičová, Jiří Mírovský, Anna Nedoluzhko, Magdaléna Rysová, Veronika Pavlíková, Jana Zdeňková, Jiří Pergler, and Radek Ocelák. Prague Discourse Treebank 1.0. Data/software, 2012b.

Poláková, Lucie, Jiří Mírovský, Anna Nedoluzhko, Pavlína Jínová, Šárka Zikánová, and Eva Hajičová. Introducing the Prague Discourse Treebank 1.0. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 91–99, Nagoya, 2013. Asian Federation of Natural Language Processing.

Poláková, Lucie, Pavlína Jínová, and Jiří Mírovský. Genres in the Prague Discourse Treebank. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 1320–1326, Reykjavík, Iceland, 2014. European Language Resources Association.

Prasad, Rashmi and Harry Bunt. Semantic relations in discourse: The current state of ISO 24617-8. In *Proceedings 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11)*, pages 80–92, 2015.

Prasad, Rashmi, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. The Penn Discourse Treebank 2.0 Annotation Manual. Technical Report IRCS-08-01, Institute for Research in Cognitive Science, Philadelphia, 2007. URL `http://www.seas.upenn.edu/~pdtb/PDTBAPI/pdtb-annotation-manual.pdf`.

Prasad, Rashmi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn Discourse Treebank 2.0. In Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 2961–2968, Marrakech, 2008. European Language Resources Association.

Roze, Charlotte, Laurence Danlos, and Philippe Muller. LEXCONN: a French lexicon of discourse connectives. *Discours. Revue de linguistique, psycholinguistique et informatique*, (10), 2012.

Rysová, Magdaléna. *Diskurzní konektory v češtině (Od centra k periferii) [Discourse Connectives in Czech (From the Centre to the Perifery)]*. PhD thesis, Charles University, Prague, Czechia, 2015.

Rysová, Magdaléna and Kateřina Rysová. The Centre and Periphery of Discourse Connectives. In Aroonmanakun, Wirote, Prachya Boonkwan, and Thepchai Supnithi, editors, *Proceedings of Pacific Asia Conference on Language, Information and Computing*, pages 452–459, Bangkok, 2014. Department of Linguistics, Faculty of Arts, Chulalongkorn University, Department of Linguistics, Faculty of Arts, Chulalongkorn University.

Rysová, Magdaléna and Kateřina Rysová. Secondary Connectives in the Prague Dependency Treebank. In Hajičová, Eva and Joakim Nivre, editors, *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 291–299, Uppsala, Sweden, 2015. Uppsala University, Uppsala University.

Rysová, Magdaléna, Pavlína Synková, Jiří Mírovský, Eva Hajičová, Anna Nedoluzhko, Radek Ocelák, Jiří Pergler, Lucie Poláková, Veronika Pavlíková, Jana Zdeňková, and Šárka Zikánová. Prague Discourse Treebank 2.0. Data/software, 2016.

Sanders, Ted JM, Wilbert PM Spooren, and Leo GM Noordman. Toward a taxonomy of coherence relations. *Discourse processes*, 15(1):1–35, 1992.

Scheffler, Tatjana and Manfred Stede. Adding Semantic Relations to a Large-Coverage Connective Lexicon of German. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Paris, France, 2016.

Schröder, Jochen. *Lexikon deutscher Präpositionen*. Verlag Enzyklopädie, 1986.

Stede, Manfred. Resolving connective ambiguity: A prerequisite for discourse parsing. *The Pragmatics of Discourse Coherence. John Benjamins, Amsterdam*, 2014.

Stede, Manfred and Yulia Grishina. Anaphoricity in Connectives: A Case Study on German. *Coreference Resolution beyond OntoNotes*, page 41, 2016.

Stede, Manfred and Arne Neumann. Potsdam Commentary Corpus 2.0: Annotation for Discourse Research. In *Proceedings of LREC 2014*, pages 925–929, Reykjavik, Iceland, 2014.

Stede, Manfred and Carla Umbach. DiMLex: A Lexicon of Discourse Markers for Text Generation and Understanding. In *Proceedings of the 17th International Conference on Computational Linguistics (Coling 1998)*, pages 1238–1242. Association for Computational Linguistics, 1998.

Synková, Pavlína, Magdaléna Rysová, Lucie Poláková, and Jiří Mírovský. Extracting a Lexicon of Discourse Connectives in Czech from an Annotated Corpus. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 1–8, Cebu, Philippines, 2017, in print. University of the Philippines Cebu.

Urešová, Zdeňka. *Valenční slovník Pražského závislostního korpusu (PDT-Vallex)*. Studies in Computational and Theoretical Linguistics. Ústav formální a aplikované lingvistiky, Praha, Czechia, 2011.

Urešová, Zdeňka, Eva Fučíková, and Jana Šindlerová. CzEngVallex: a bilingual Czech-English valency lexicon. *The Prague Bulletin of Mathematical Linguistics*, 105:17–50, 2016.

Veselovská, Kateřina and Ondřej Bojar. Czech SubLex 1.0, 2013.

Zeman, Daniel, David Mareček, Jan Mašek, Martin Popel, Loganathan Ramasamy, Rudolf Rosa, Jan Štěpánek, and Zdeněk Žabokrtský. HamleDT 3.0, 2015.

Zeyrek, Deniz and Murathan Kurfalı. TDB 1.1: Extensions on Turkish Discourse Bank. *LAW XI 2017*, page 76, 2017.

Zeyrek, Deniz, Işin Demirşahin, Ayişiği Sevdik-Çalli, Hale Ögel Balaban, İhsan Yalçinkaya, and Ümit Deniz Turan. The annotation scheme of the Turkish Discourse Bank and an evaluation of inconsistent annotations. In *Proceedings of the fourth Linguistic Annotation Workshop*, pages 282–289, 2010.

Zhou, Yuping and Nianwen Xue. PDTB-style discourse annotation of Chinese text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 69–77, 2012.

Zhou, Yuping and Nianwen Xue. The Chinese discourse treebank: a Chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397, 2015.

Zikánová, Šárka, Eva Hajičová, Barbora Hladká, Pavlína Jínová, Jiří Mírovský, Anna Nedoluzhko, Lucie Poláková, Kateřina Rysová, Magdaléna Rysová, and Jan Václ. *Discourse and Coherence. From the Sentence Structure to Relations in Text*. Studies in Computational and Theoretical Linguistics. ÚFAL, Praha, Czechia, 2015.

Address for correspondence:
Jiří Mírovský
`mirovsky@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

The Prague Bulletin of Mathematical Linguistics
NUMBER 109 OCTOBER 2017

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.