The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017

EDITORIAL BOARD

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

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

ISSN 0032-6585

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017

CONTENTS

Articles

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017 5–17

Neural Monkey: An Open-source Tool for Sequence Learning

Jindřich Helcl,[a b] Jindřich Libovický[a]

[a] Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics
[b] German Research Center for Artificial Intelligence (DFKI), Language Technology Lab

Abstract

In this paper, we announce the development of Neural Monkey – an open-source neural machine translation (NMT) and general sequence-to-sequence learning system built over the TensorFlow machine learning library. The system provides a high-level API tailored for fast prototyping of complex architectures with multiple sequence encoders and decoders. Models' overall architecture is specified in easy-to-read configuration files. The long-term goal of the Neural Monkey project is to create and maintain a growing collection of implementations of recently proposed components or methods, and therefore it is designed to be easily extensible. Trained models can be deployed either for batch data processing or as a web service. In the presented paper, we describe the design of the system and introduce the reader to running experiments using Neural Monkey.

1. Introduction

Neural machine translation (NMT) recently became a new successful paradigm in machine translation (Sutskever et al., 2014; Bahdanau et al., 2014). Besides NMT, sequence-to-sequence (S2S) learning in general proved its usefulness in various other tasks including building a chatbot (Vinyals and Le, 2015), image captioning (Vinyals et al., 2015; Xu et al., 2015), or text segmentation and entity recognition (Gillick et al., 2016).

Neural Monkey is an open-source toolkit written using the TensorFlow machine learning library (Abadi et al., 2016). It provides a higher level API, such that it should be enough for the users to be familiar with the models on the equation level, without delving into implementation details. For that reason, we try to use as big abstract

 Corresponding author: helcl@ufal.mff.cuni.cz
Cite as: Jindřich Helcl, Jindřich Libovický. Neural Monkey: An Open-source Tool for Sequence Learning. The Prague Bulletin of Mathematical Linguistics No. 107, 2017, pp. 5–17. doi: 10.1515/pralin-2017-0001.

building blocks as possible. Unlike *tfLearn*[1] or *Lasagne*,[2] our building blocks are not individual network layers, but more abstract objects like encoders or classifiers. These objects are parametrized so that their properties (e.g. number and sizes of hidden layers or dropout probability) can be set from a farther perspective. This design decision allows us to place the configuration of the experiments away from the actual code into a separate comprehensive configuration file. This way, users are prevented from interleaving the configuration with other program logic.

Neural Monkey is still under development and has an ambition to become an ever-growing collection of recent innovations in NMT and S2S learning in general, which would allow its users to easily try out the model for their specific tasks and datasets. With its simple experiment management, we hope that it will be used as a ready-made easily-extensible toolkit for experiments with machine translation, image captioning, text classification tasks or scene text recognition. It has already been used for the submission for the WMT16 automatic post-editing and multimodal translation tasks (Libovický et al., 2016), linguistic analysis of MT systems (Avramidis et al., 2016).

The current version of the software is available at `https://github.com/ufal/neuralmonkey` under the BSD license.

2. Related Work

Most of the deep learning frameworks split the computation into two stages. In the first stage, the programmer designs the computation symbolically. The symbolic code which describes a computation graph is then optimized and compiled to efficiently perform the numeric computation. In the second stage, the program performs the numerical computation on the compiled computation graph using a supplied input data.

In Theano (Bergstra et al., 2010), C code is generated from the original Python code and then it is compiled. One of the problems with this approach is that it is difficult to track the computation once the code is compiled.

Chainer (Tokui et al., 2015), on the other hand, is written in Python, and therefore, debugging of the code is easier. In order to match the speed of compiled C code, Chainer uses fast libraries for numerical computation both on CPUs and GPUs.

TensorFlow (Abadi et al., 2016) stays in between of these approaches. Its building blocks are implemented in C and are manipulated using Python code. Additionally, TensorFlow provides tools for greater amount of transparency and offers more convenient ways of debugging the computation graphs.

Torch (Collobert et al., 2011) also uses a similar approach, employing a scripting language that operates over a highly optimized C backend. Unlike the other frameworks which use Python, Torch uses Lua for the scripts.

[1] `https://github.com/tflearn/tflearn`

[2] `https://github.com/Lasagne/Lasagne`

These libraries are basically general purpose tools for linear algebra computations (although they bring more and more functionality tailored for deep learning) and writing the models requires typing a lot of service code that deals with preparing the inputs and outputs of the models. All of these three basic libraries are distributed under free and commercial-friendly licenses.

For the reasons above, various libraries provide higher levels of abstractions over the basic libraries. Most of them aim for general-purpose use and do not provide many features tailored for natural language processing. Among these, there are the previously mentioned *tfLearn* and *Lasagne*, built on top of TensorFlow and Theano, respectively. Another popular library for Theano, similar to *Lasagne*, is *Blocks* (van Merriënboer et al., 2015). All of these libraries offer a lower level abstraction, operating directly with layers. *Keras* (Chollet, 2015) operates with the layers similarly as the other frameworks. Currently, it provides probably the richest set features both over Theano and TensorFlow.

A software package similar to Neural Monkey is *Nematus* (Sennrich et al., 2017). It is a state-of-the-art research software for NMT. Unlike Neural Monkey, it is written using Theano and focuses only on single-input and single-output setup and does not by default support input of non-textual modalities.

Another software package for NMT is *OpenNMT* (Klein et al., 2017), which is built on Torch. Similarly to Nematus, it focuses on single-source and single-output models. Besides that, there is an extension for image-to-text models. Compared to Nematus, it does not support so many features for NMT such as sub-word units or direct optimization towards BLEU score (Shen et al., 2016).

Unlike Neural Monkey, none of the S2S learning packages above provides basic experiment management functionality.

3. Problem Conceptualization

This section provides the basic overview on how Neural Monkey conceptualizes the problem of S2S learning and how the data flows during training and how running the models looks like.

3.1. Loading and Processing Datasets

Before the models are applied to the data, there is a short pipeline of steps preparing the data. For that, we introduce the notion of dataset and data series.

A *dataset* is a collection of named data *series*. Each data series is a list of data instances of the same type that represents a single kind of input or desired output of a model. In the simple case of machine translation, there are two data series: a list of sentences in the source language and a list of sentences in the target language. Figure 1 captures how a dataset is created from the input data.

The dataset is created in the following steps:

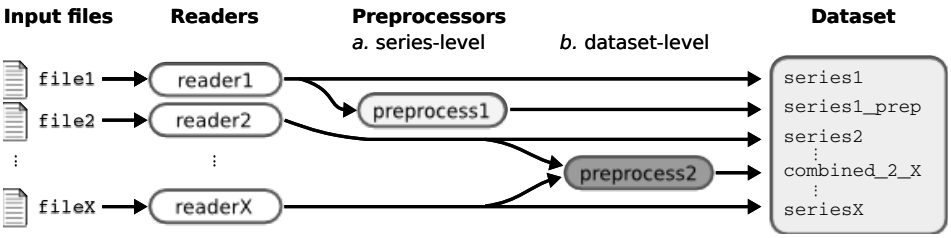


Figure 1. Creating a dataset in Neural Monkey

1. An input file is read using a reader. Reader can e.g., load a file containing paths to JPEG images and load them as NumPy arrays, or read tokenized text as a list of lists (sentences) of string tokens.
2. Series created by the readers can be preprocessed by some series-level preprocessors. An example of such preprocessing is byte-pair encoding (Sennrich et al., 2016) which loads a list of merges and segments the text accordingly.
3. The final step before creating a dataset is applying dataset-level preprocessors which can take more series and output a new series.

Currently, there are two implementations of a dataset. An in-memory dataset which stores all data in the memory, and a lazy dataset which gradually reads the input files step by step and only stores the batches necessary for the computation in the memory.

3.2. Training and Running a Model

This section describes the training and running workflow. In general, we try to separate model parts which provide a declarative description of the model (e.g., equations for an RNN decoder) and the way the models are run (e.g., run it using a beam search, sample a random sentence, or get the error derivatives). The main concepts and their interconnection can be seen in the scheme in Figure 2. It shows how the model is executed on the data using what we call *runners*.

Dataset series can be used to create a *vocabulary*. A vocabulary represents an indexed set of tokens and provides functionality for converting lists of tokenized sentences into matrices of token indices and vice versa. Vocabularies are used by encoders and decoders for feeding the provided series into the neural network.

The model itself is defined by model parts which categorize into encoders and decoders. This is where most of the TensorFlow code is located. Encoders are parts of the model which take some input and compute a representation of it. Decoders are model parts that produce some outputs. Our definition of encoders and decoders is more general than in the classical S2S learning. An encoder can be for example a convolutional network processing an image. The RNN decoder is for us only a special

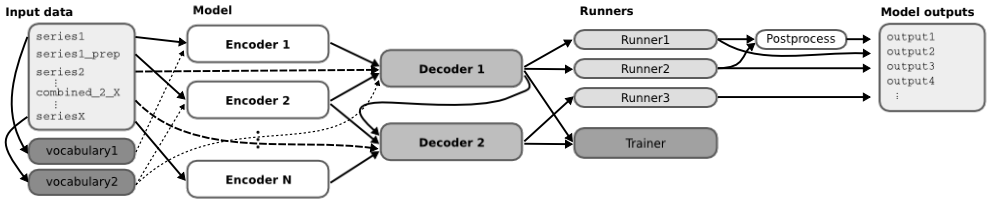


Figure 2. Model workflow in Neural Monkey

type of decoder, it can be also a sequence labeler or a simple multilayer perceptron classifier or regressor.

Decoders are executed using so-called *runners*. Different runners represent different ways of running the model. We might want to get a single best estimation, get an n-best list or a sample from the model. We might want to use an RNN decoder to get the decoded sequences or we might be interested in the word alignment obtained by its attention model. This is all done by employing different runners over the decoders. The outputs of the runners can be subject to further postprocessing.

In addition to runners, each experiment has to have its *trainer*. A trainer is a special case of a runner that actually modifies the parameters of the model. It collects the objective functions and uses them in an optimizer.

Neural Monkey manages TensorFlow sessions using an object called TensorFlow manager. Its basic capability is to execute runners on provided datasets. It encapsulates all logic concerning the TensorFlow sessions.

We can demonstrate this rather abstract description on an example of automatic post-editing of MT output. The model is illustrated in Figure 3. In this case, the model uses two encoders which are implemented using a bi-directional GRU (Cho et al., 2014) network. The first encoder is fed with sentences in the source language and the second one is fed with the machine translation output. A decoder combines outputs of both the encoders and attends to their hidden states during decoding. A trainer that follows the decoders computes the error of the decoder outputs given the desired target text from the train dataset.

4. Configuration

The experiments are described using configuration files. They contain a complete specification of the network architecture, preprocessing and postprocessing of the data, the training and validation data, all meta-parameters of the training, and metrics used for model evaluation. The configuration files are the main tool of Neural Monkey's experiment management. The same configuration can be used after training to run the trained model.

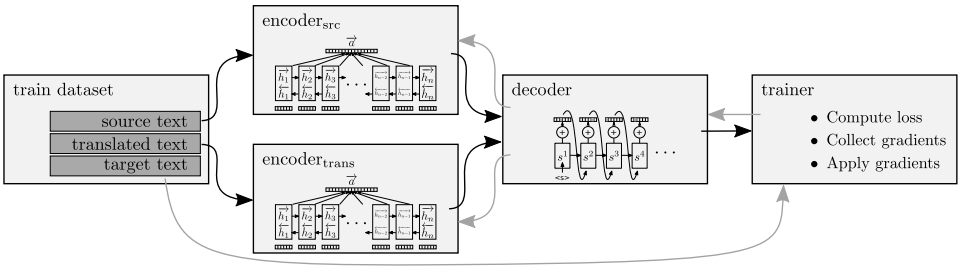


Figure 3. Scheme of the architecture for the training process of an automatic post-editing task

The following sections explain the main ideas how the configuration works.

4.1. Syntax

The configuration files are based on the syntax of INI files.[3] Neural Monkey INI files contain key-value pairs, delimited by an equal sign (=) with no spaces around. The key-value pairs are grouped into sections. (Neural Monkey requires all pairs to belong to a section.)

Every section starts with its header which consists of the section name in square brackets. Everything below the header is considered to be a part of the section. Comments can appear on their own line, prefixed either with a hash sign (#) or a semicolon (;) and possibly indented. The configuration introduces several additional constructs for the values. These can be atomic values, and compound values.

The supported atomic values are:

- booleans: literals `True` and `False`;
- integers: strings that could be interpreted as integers by Python (e.g., `1`, `002`);
- floats: strings that could be interpreted as floats by Python (e.g., `1.0`, `.123`, `2.`, `2.34e-12`);
- strings: string literals in quotes (e.g., `"walrus"`, `"5"`);
- section references: string literals in angle brackets (e.g., `<encoder>`), sections are later interpreted as Python objects;
- Python names: strings without quotes which are neither booleans, integers and floats, nor section references (e.g., `neuralmonkey.encoders.SentenceEncoder`).

On top of that, there are two compound types syntax from Python:

- lists: comma-separated in squared brackets (e.g., `[1, 2, 3]`);

[3] `https://en.wikipedia.org/wiki/INI_file`

- tuples: comma-separated in round brackets (e.g., `("target", <ter>)`).

4.2. Interpretation

Each configuration file contains a `[main]` section which is interpreted as a dictionary having keys specified in the section and values which are results of interpretation of the right-hand sides.

Both the atomic and compound types taken from Python (i.e., everything except the section references) are interpreted as their Python counterparts.

Section references are interpreted as references to objects constructed while interpreting the referenced section. (So, if `<session_manager>` is on the right-hand side of a variable assignment and section `[session_manager]` is located elsewhere in the file, Neural Monkey will construct a Python object based on the key-value pairs in section `[session_manager]` and use it as a value for the variable.)

Each section, except for the `[main]` section, needs to contain the key `class` with a value of Python name which is a callable (e.g., a class constructor or a function). The other keys are used as named arguments of the callable.

4.3. Content of the Configuration

The configuration needs to specify what is needed during the model runtime (the model architecture, preprocessing and postprocessing of the data) and the configuration for training (loss computation, mini-batch sizes etc.). Since the configuration refers directly to the Python code, there is no need for separate user and programmer documentation. The documentation of the configuration elements can be found in the API documentation of the respective modules of the package. A few self-explanatory examples of the configuration can be seen in Figures 4, 5, and 6.

5. Usage

Neural Monkey is written in Python 3.5. Because of the abstraction over computing devices that TensorFlow offers, the models can be run both on CPU and GPU. Neural Monkey has only a few requirements (besides TensorFlow), all of them can be easily installed with pip.

There are four main scripts that are used to run Neural Monkey. They are located in the `bin` directory of the source repository:

- `neuralmonkey-train` – The main script used for training a model. It takes one argument, which is the location of the configuration file for the training. Each experiment has its own directory which contains a copy of the configuration file, the current git diff and commit ID, all experiment logs, TensorBoard visualization files, and the saved model.

```
[encoder]
class=(...).SentenceEncoder
name="encoder-1"
rnn_size=256
max_input_len=50
embedding_size=200
dropout_keep_prob=0.5
attention_type=(...).Attention
data_id="source"
vocabulary=<encoder_vocabulary>
```

```
[decoder]
class=(...).Decoder
name="decoder"
encoders=[<encoder>]
rnn_size=256
max_output_len=50
embedding_size=256
use_attention=True
dropout_keep_prob=0.5
data_id="target"
vocabulary=<decoder_vocabulary>
```

```
[trainer]
class=(...).CrossEntropyTrainer
decoders=[<decoder>]
l2_regularization=1.0e-8
```

```
[runner]
class=(...).GreedyRunner
decoder=<decoder>
output_series="target"
```

Figure 4. An example of construction of the encoder, decoder, trainer and runner objects by direct calls of their class constructors.

```
[train_data]
class=dataset.load_dataset_from_files
s_source="tests/data/train.tc.en"
s_target="tests/data/train.tc.de"
s_target_out="train.translated.de"
```

The `dataset.load_dataset_from_files` function is called as the dataset initializer. All arguments starting with `s_` correspond to named data series (`source` and `target`) and provide paths to files containing the data. If a data series argument ends with `_out`, the path is interpreted as the output file for the series.

Figure 5. Example of a dataset. The <val_data> *object is created analogically.*

```
[encoder_vocabulary]
class=vocabulary.from_dataset
datasets=[<train_data>]
series_ids=[source]
max_size=25000
save_file="enc_vocab.pkl"
```

This snippet shows how a vocabulary is created from a dataset. The function simply iterates through the dataset series `source`, computes the tokens frequency and keeps 25,000 most frequent tokens. After the vocabulary is created, it is stored in the specified file.

Figure 6. Example of creating vocabulary. The <decoder_vocabulary> *object is created analogically.*

- `neuralmonkey-run` – This script is used for loading a trained model and its execution on a dataset. It requires the location of the original configuration file and another configuration file that specifies the location of the dataset files.
- `neuralmonkey-server` – A trained model can be run as a web service. It accepts dictionaries of data series for encoders and returns the series produced by decoders in a JSON format with a REST API.
- `neuralmonkey-logbook` – The experiment logs and configurations can be browsed using a small web service Neural Monkey LogBook. It is called with a `--logdir` argument, which points to a directory with experiments. When the directory has sub-directories, they are regarded as separate experiments and are all shown in the LogBook.

6. Implemented Features

This section provides a short overview on features that have been already implemented in Neural Monkey:

- standard attentive S2S learning with a decoder capable of working with multiple encoders with GRU or LSTM networks;
- sequence labeling and sequence classification and regression;
- pre-trained ImageNet network for experiments with image captioning and multimodal machine translation (with optional model fine-tuning);
- custom deep convolutional networks for image processing;
- beam search and model ensembling;
- conditional gated recurrent units for decoder (Firat and Cho, 2016);
- byte-pair encoding for translation with sub-word units (Sennrich et al., 2016).

The possibility to use multiple decoders at one moment allows to easily build architectures for multi-task learning, like joint training of POS tagging on the source language together with machine translation.

Other advanced features like layer normalization or various learning methods optimizing the outputs directly towards BLEU score (Shen et al., 2016; Rennie et al., 2016) are expected to be added in the future.

7. Benchmarking

In order to validate the Neural Monkey's performance, we a sanity check evaluation on the architecture introduced by Bahdanau et al. (2014) which became the standard baseline model in NMT research.[4] Following Bahdanau et al. (2014), we train and evaluate our models using data provided for the WMT14 news task[5] from

[4]Reference implementation using Theano can be found here: `https://github.com/lisa-groundhog/GroundHog`

[5]Task details can be found here: `http://www.statmt.org/wmt14/translation-task.html`

model	BLEU score	epochs
Bahdanau et al. (2014) – neural	26.75	2.2
Bahdanau et al. (2014) – neural	28.45	6.0
SMT	33.30	—
Neural Monkey – greedy decoding	25.08	1.2
Neural Monkey + CGRU – greedy decoding	27.35	1.6

Table 1. Results achieved by Neural Monkey on the WMT14 News Task French to English dataset with the number epochs the training was running.

English to French. We use the same dataset as Bahdanau et al. (2014) for both training and evaluation of the models.

The encoder is a bi-directional GRU network (Cho et al., 2014) with 500 hidden units in each direction, the decoder is an RNN decoder with 1,000 units in the hidden layer. Unlike the original model, we do not use the max-out projection, but a simple 'tanh' projection. Whereas the original paper used Adadelta (Zeiler, 2012) optimizer, we used Adam (Kingma and Ba, 2014) in our experiments. The comparison of the models is shown in Table 1.

We have also experimented with conditional GRU (Firat and Cho, 2016) units which show the expected improvement on top of the baseline model. Another easily achievable improvement using Neural Monkey could be done with sub-word units (Sennrich et al., 2016) to deal with the out-of-vocabulary tokens.

8. Conclusions

We presented *Neural Monkey* – a new toolkit for sequence learning built using TensorFlow. It is aimed at gathering implementations of recent deep learning methods in various fields, primarily focused on NMT. Compared to other toolkits, it provides a higher level of abstraction, along with a simple configuration mechanism that allows for fast prototyping and reusing trained models and experiment management.

In the future, more features will be added. We also hope for community feedback from using the toolkit in practice that will help us to reevaluate the design decisions we have made. For the information on the recent development, we refer the reader to the online documentation. [6]

Acknowledgments

This research was supported by the Charles University grant no. GAUK 52315/2015, SVV 260 224, EU grants no. FP7-ICT-2013-10-610516 (QTLeap), H2020-ICT-2014-

[6] `https://readthedocs.org/neuralmonkey`

1-645452 (QT21), and H2020-ICT-2014-1-644753 (KConnect), and the Czech Science Foundation grant no. P103/12/G084 (CEMI).

Bibliography

Abadi, Martın, 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.

Avramidis, Eleftherios, Vivien Macketanz, Aljoscha Burchardt, Jindrich Helcl, and Hans Uszkoreit. Deeper Machine Translation and Evaluation for German. In Hajič, Jan, Gertjan van Noord, and António Branco, editors, *Proceedings of the 2nd Deep Machine Translation Workshop*, pages 29–38, Praha, Czechia, 2016. ÚFAL MFF UK, ÚFAL MFF UK. ISBN 978-80-88132-02-8. URL `http://www.aclweb.org/anthology/W16-6404`.

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/1410.0473`.

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.

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-4012`.

Chollet, François. Keras. `https://github.com/fchollet/keras`, 2015.

Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

Firat, Orhan and Kyunghyun Cho. Conditional Gated Recurrent Unit with Attention Mechanism. https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf, May 2016. Published online, version `adbaeea`.

Gillick, Dan, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. Multilingual Language Processing From Bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-1155`.

Kingma, Diederik P. and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *CoRR*, abs/1701.02810, 2017. URL `http://arxiv.org/abs/1701.02810`.

Libovický, Jindřich, Jindřich Helcl, Marek Tlustý, Pavel Pecina, and Ondřej Bojar. CUNI System for WMT16 Automatic Post-Editing and Multimodal Translation Tasks. *CoRR*, abs/1606.07481, 2016. URL http://arxiv.org/abs/1606.07481.

Rennie, Steven J., Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical Sequence Training for Image Captioning. *CoRR*, abs/1612.00563, 2016. URL http://arxiv.org/abs/1612.00563.

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, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain, 2017.

Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum Risk Training for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1159.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Tokui, Seiya, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: A next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

van Merriënboer, Bart, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and Fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015. URL http://arxiv.org/abs/1506.00619.

Vinyals, Oriol and Quoc V. Le. A Neural Conversational Model. In *ICML Deep Learning Workshop*, 2015. URL http://arxiv.org/pdf/1506.05869v3.pdf.

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164, 2015. doi: 10.1109/CVPR.2015.7298935. URL http://dx.doi.org/10.1109/CVPR.2015.7298935.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. 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 2015*, pages 2048–2057, Lille, France, 2015. URL http://jmlr.org/proceedings/papers/v37/xuc15.html.

Zeiler, Matthew D. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

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

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017 19–38

Difference between Written and Spoken Czech: The Case of Verbal Nouns Denoting an Action

Veronika Kolářová,[a] Jan Kolář,[b] Marie Mikulová[a]

[a] Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics
[b] Institute of Mathematics of the Czech Academy of Sciences

Abstract

The present paper extends understanding of differences in expressing actions by verbal nouns in corpora of written vs. spoken Czech, namely in the Czech part of the Prague Czech-English Dependency Treebank and in the Prague Dependency Treebank of Spoken Czech.

We show that while the written corpus includes more complex noun phrases with more explicit expression of adnominal participants, noun phrases in the spoken corpus contain more deletions and more exophoric references. We also carried out a quantitative analysis focusing on relative frequencies of combinations of participants modifying verbal nouns; although the written corpus shows higher relative frequencies, the order of the relative frequencies of particular combinations is the same in both types of communication.

1. Introduction

Differences between written and spoken language have come under scrutiny in linguistic research in English and other languages including Czech for decades (e.g., Halliday, 1967; Hausenblas, 1962; Chafe and Danielewicz, 1987). Though older studies were based on authentic spoken examples or even on collections of spoken texts compiled for the particular purpose, a new dimension of research of spoken language has been added by the development of large corpora of spoken communication. From the linguistic point of view, however, only few of the resources are POS tagged and/or lemmatized, or even include syntactic annotation. The Prague Dependency Treebank of Spoken Czech (which is the resource for our research, see Section 3), the Switchboard corpus in the Penn Treebank-3 (Marcus et al., 1999), Childes Database

 Corresponding author: kolarova@ufal.mff.cuni.cz
Cite as: Veronika Kolářová, Jan Kolář, Marie Mikulová. Difference between Written and Spoken Czech: The Case of Verbal Nouns Denoting an Action. The Prague Bulletin of Mathematical Linguistics No. 107, 2017, pp. 19–38.
doi: 10.1515/pralin-2017-0002.

	Written communication	**Spoken communication**
Expression	condensed / complex / intricate sentences	analytical / unelaborated flow of speech
Specific means of expression	hypotaxis, nominalisations	parataxis, repetitions, restarts, corrections, disfluencies
Segmentation	strict / clear boundaries between sentences	unclear sentence segmentation, juxtaposition
Deletions / ellipses	deletions / ellipses with context-dependent references	incompleteness, fragmentation, interruptions, extra-textual (exophoric) references
Valency	refinement of forms of participants	marked participants and forms

Table 1. Main differences in syntax between written and spoken communication

(MacWhinney, 2000; Sagae et al., 2004), or Corpus Gesproken Nederlands (Schuurman et al., 2003) are among the few exceptions to this rule.

In spite of this situation, the data-based research in various aspects of spoken language has recently become one of the central topics (e.g., Biber et al., 1999; Brazil, 1995; Roberts and Street, 1997; Leech, 2000). Within the spoken Czech research, which mostly happens on the national scene, real texts and dialogues are analysed and presented (e.g., Těšitelová, 1983), with focus on the specificity of the spoken word form (e.g., Šonková, 2008; Cvrček et al., 2010), spoken syntax (e.g., Müllerová, 1994; Hoffmannová, 2012; Mikulová and Hoffmannová, 2011), issues of valency (e.g., Mikulová et al., 2013) and the specificity of the social issues concerning the speakers and situations in which the analysed utterances were used (e.g., Hoffmannová et al., 1999; Hoffmannová and Müllerová, 2007; Čmejrková et al., 2004). Despite the numerous studies, the description of syntax of spoken Czech is still not as developed, consistent and comprehensive as the description of written Czech. This article aims at a description of differences between the written and spoken syntax, focusing on a special case of action-denoting verbal nouns in corpora of written and spoken Czech.

2. Differences between written and spoken language

On the basis of the studies mentioned in Section 1, we summarize the main differences in syntax between both types of communication (see also Table 1).

A prominent feature of spoken language besides its acoustic nature is its anchoring in the time and situation. The conditions of spontaneous speech production (presence of the addressee, speaking skills of the speakers, importance of non-verbal communication) lead to numerous repetitions, incomplete sentences, corrections and interruptions. Presence of the addressee, context and knowledge shared by the speakers play an important role, so it is possible to leave much unsaid or indirectly implied in the spontaneous speech. On the other side, writers receive no immediate feedback from their readers so there is more need to explain things clearly and unambiguously than in speech. Using longer sentences and many subordinate clauses, written texts are usually more complex and intricate than speech.

Verbal nouns denoting an action belong to nominalisations and they can be understood as reclassifications of their corresponding verbal clauses (Heyvaert, 2003). They help to form compact and condensed expression and when they are modified by their participants, they constitute complex noun phrases.[1] We suppose that a written text includes more complex noun phrases with more explicit expression and that noun phrases in spontaneous speech contain more deletions, using more exophoric references. We test this hypothesis in corpora of written and spoken Czech containing deep syntactic annotation (see Section 3).

3. Data: Corpora of written and spoken Czech with deep syntactic annotation

The syntactic behaviour of Czech verbal nouns can be studied most effectively in syntactically annotated corpora. One of the features characteristic of syntax of spoken language is its incompleteness and fragmentation (Hunyadi, 2013). We are convinced that the unexpressed elements should become an important part of the research of the differences between written and spoken communication. However, as elements that are not present on the surface layer of a sentence, their reconstruction relies on a theoretical framework that deals with the deep structure of sentences and therefore they are only exceptionally captured even in syntactically annotated corpora. In order to be able to search for the unexpressed elements in the syntactic structure of written and spoken communication, we use manually syntactically annotated corpora built within the theoretical framework of Functional Generative Description (FGD, Sgall et al., 1986) because they capture also deletions (see Section 4).

Further aspect we considered when selecting resources that best meet our requirements is the need of comparable data. The data should be comparable especially in its annotation scheme. Thus we chose two corpora from the Prague Dependency Treebank family which have comparable size and, moreover, which apply the same guidelines for annotation of valency of verbal nouns. These two corpora are (i) the

[1] In this paper, we use the term noun phrase for nominal constructions in which a noun is modified by its dependents, focusing on verbal nouns modified by their participants.

	PCEDT (written corpus)	PDTSC (spoken corpus)
Tokens	1 162 072	742 257
Sentences	49 208	73 835
Words per a sentence	23.6	10.1
Content verbs	99 186	102 868

Table 2. Comparison of the size of the used written and spoken corpora

Prague Dependency Treebank of Spoken Czech (PDTSC), and (ii) the Czech part of the Prague Czech-English Dependency Treebank (PCEDT). The unique opportunity of having a spoken and written resource with a comparable annotation enables us to carry out precise and reliable analysis of selected differences between the two types of communication.

(i) **The Prague Dependency Treebank of Spoken Czech 2.0** (PDTSC) is the upcoming release (planned to be published in 2017; Mikulová et al., in press).[2] The corpus offers a huge, unique material for a systematic analysis of syntax of spoken Czech on higher levels of linguistic abstraction, including deep syntactic annotation. PDTSC recordings consist of two types of dialogues. First, it contains slightly moderated testimonies of Holocaust survivors from the Malach project corpus.[3] The second part of the corpus consists of dialogues recorded for the Companions project.[4] It contains personal memories, but in a setting where the two dialogue participants chat over a collection of photographs. The spoken material is manually transcribed, edited for disfluencies, and then annotated syntactically (on the layer of surface syntax and deep syntax) while keeping the original transcript explicitly aligned with the edited version. This allows the morphological, syntactic and semantic annotation to be deterministically and fully mapped back to the transcript and audio. The PDTSC consists of 742 257 tokens and 73 835 sentences, representing 6 174 minutes of spontaneous dialogue speech.

(ii) **The Prague Czech-English Dependency Treebank 2.0** (PCEDT, Hajič et al., 2012) is a manually parsed Czech-English parallel corpus of 1.2 million tokens in 49 208 sentences for each language. The English part holds the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1999). The Czech part was translated from the English source sentence by sentence.

[2]The results of our search can differ from the future published corpus but the differences will be insignificant (because there are not a lot of substantial changes in the data now).

[3]`http://malach.umiacs.umd.edu/`

[4]`http://www.companions-project.org`

Table 2 shows that these two corpora are comparable in size (cf. the number of tokens) but it also reflects some differences between the written and spoken communication, especially the difference in number of sentences and their length (there are more sentences in the spoken corpus but the sentences are shorter on average than in the written corpus). Searching through the two corpora is carried out by the tool called PML-TQ (Štěpánek and Pajas, 2010).

We are aware that the two corpora are not representative samples of written and spoken communication. The Czech part of the PCEDT is a translation rather than an original written text, moreover it focuses on a very specialized semantic domain concerning trading. Neither our spoken corpus, PDTSC, contains completely spontaneous (i.e., unprepared) spoken production. However, despite these deficiencies, we take advantage of their deep syntactic annotation and show that even such imperfect samples of written and spoken communication reflect significant differences in expressing an action by verbal nouns. On the basis of the two corpora, we can specify the following distinctive features of denoting an action by verbal nouns in Czech written and spoken communication: degree of compact and condensed expression measured by frequency of verbal nouns (Section 6.1), noun phrase complexity (Section 6.2), degree of explicitness (Section 6.3), and finally deletions and exophoric references (Section 6.4).

4. Prague Dependency Treebank family: Annotation scheme

4.1. Valency

As mentioned above, one of the important features of the PDT-style annotation (Hajič et al., in press) is the fact that in addition to the morphological layer and to the syntactic annotation of the surface shape of the sentences the scenario includes a complex semantically based annotation on the highest, deep syntax layer (so-called tectogrammatical layer). The core component in the annotation is valency and one of the important features is the reconstruction of surface deletions on the tectogrammatical layer (the annotation guidelines are formulated in Mikulová et al., 2006; Mikulová, 2014). The valency theory for the theoretical framework of the FGD was formulated by Panevová (1974, 1975) and it has been detailed in numerous studies addressing especially valency of verbs (Panevová, 1998, 1999, 2014) and nouns (Piťha, 1980; Panevová, 2002; Kolářová, 2014). The following types of complementations (i.e. the individual dependency relations) are able to fill in the individual slots of the valency frames of verbs:

– inner participants or arguments that can be obligatory or optional: Actor, Patient, Addressee, Effect, Origin (e.g., *Vláda omezila těžbu uranu ze současných 950 tun na 500 tun ročně* 'The government restricted uranium mining from the current 950 tonnes to 500 tonnes per year');

– obligatory free modifications or adjuncts, especially those with the meaning of direction (e.g., *přijet někam* 'to arrive somewhere') or location (e.g., *přebývat někde* 'to dwell somewhere') and manner (e.g., *chovat se dobře* 'to behave well').

Within the concept of nominal valency in the framework of the FGD, the same repertoire of valency complementations is assumed for deverbal nouns denoting an action. The repertoire of valency complementations of non-deverbal nouns and deverbal nouns undergoing substantial shifts in their meaning is supplemented with some more modifications, especially with a special nominal participant Material (e.g., *skupina lidí* 'group of people', *jedno balení másla* 'one package of butter') and a free modification Appurtenance (e.g., *Petrovo auto* 'Peter's car', *oddělení odbytu* 'sales department').

The valency theory was applied to the PDT-corpora data which resulted in a very complex and detailed annotation scheme. Different meanings of words with valency that occur in the data are differentiated in a valency lexicon called PDT-Vallex[5] (Hajič et al., 2003; Urešová, 2012). Each PDT-Vallex entry describes a lexeme (represented by the "lemma") and its valency frame(s). One valency frame typically corresponds to one meaning (sense) of a word (i.e., a verb, a noun, an adjective, or an adverb). Although PDT-Vallex does not explicitly work with the term lexical unit, a meaning of a word with its particular valency frame corresponds to a lexical unit, understood roughly as 'a given word in a given sense' (Cruse, 1986). In PDT-Vallex, a valency frame encodes the core valency information, listing possible alternative forms of valency complementations and giving information about semantic roles, i.e., deep functions in terms of tectogrammatical functors of the FGD, esp. ACT for Actor, PAT for Patient, ADDR for Addressee, ORIG for Origin or EFF for Effect. Moreover, information about obligatoriness is assigned to each participant (optional participants are marked with the sign '?', see (1) and (2) in Section 5) and it is reflected in the deep structure of sentences in which the respective noun occurs as follows: nodes for valency complementations that are obligatory and thus present in the deep structure of the sentence are added into the data even though they are not present on the surface layer of the sentence. This is exactly where ellipsis meets valency: an unexpressed obligatory participant or free modification is treated as a surface deletion (valency ellipsis) and it is captured by adding a node to the tree (for more details concerning coreference types see Section 4.2). Nodes added for obligatory complementations that are not present on the surface layer of the sentence enable us to search for the unexpressed elements which we consider crucial for our research into the differences between written and spoken communication.

To summarize, the annotation of valency in the PDT-corpora consists of:

– determining and assigning a valency frame from PDT-Vallex;
– a corresponding semantic role (ACT, PAT, ADDR, etc.) is assigned to the nodes for valency complementations expressed on surface;

[5] http://hdl.handle.net/11858/00-097C-0000-0023-4338-F

– obligatory valency complementations unexpressed on the surface are captured by an added (newly created) node with an artificial lemma (for example #PersPron), and the corresponding semantic role is also assigned.

4.2. Coreference relations

The PDT-style annotation also captures various types of (co)reference relations. For each participant (when not expressed on the surface, it is captured by an added node, see Section 4.1), the annotator determines whether the participant has its antecedent in the text (core coreference relations) or whether there is a reference to a situation or reality external to the text (exophoric references), or whether there is no (co)reference. Within the core coreference relations, the two following types are distinguished: grammatical coreference (in which it is possible to pinpoint the antecedent according to grammatical rules) and textual coreference (where reference is determined not only by grammatical means, but also via context). (Co)reference relations are annotated in the case of personal and possessive pronouns, demonstrative pronouns *ten, ta, to* 'this/that', and in the case of unexpressed obligatory participants. Various types of coreference relations are captured by assignments of artificial lemmas of various types and via various types of coreference arrows from the participant (coreferring node) to its antecedent. An exophoric reference is represented as a short arrow pointing upwards (for more details see Zikánová et al., 2015).

Depending on the type of (co)reference relations, we distinguish the following types of obligatory participants of verbal nouns denoting an action (some of them are addressed in Section 6.3 and Section 6.4):

(a) a participant expressed by a noun, a possessive adjective, a prepositional phrase, a content clause or an infinitive (e.g., *Petrovo*.ACT *pití čaje*.PAT 'Peter's drinking of tea')

(b) a participant expressed by a pronoun (e.g., *jeho*.ACT *pití toho*.PAT 'his drinking of that') with one of the following types of (co)reference:
  (ba) grammatical coreference
  (bb) textual coreference
  (bc) exophoric reference
  (bd) no reference (in the case of idioms, e.g., *mít své opodstatnění*, lit. *to have its justification*, i.e., 'be justifiable')

(c) a participant unexpressed on the surface (e.g., *pití* 'drinking') with one of the following types of (co)reference:
  (ca) grammatical coreference
  (cb) textual coreference
  (cc) exophoric reference
  (cd) no reference (in the case of the so-called general participant, e.g., *tuk na smažení* 'frying fat').

5. Verbal nouns denoting an action

In this paper, we concentrate on expressing an action by Czech verbal nouns which are derived from verbs by productive means (suffixes *-(e)ní/tí*, as in *vykládání* 'explaining//unloading' or *pojetí* 'conception'). We do not consider another type of Czech deverbal nouns that also in some of their meanings denote an action, i.e., nouns derived from verbs by non-productive means including the zero suffix (such as *vykládka* 'unloading', *výklad* 'explanation/interpretation'). There are three reasons for working only with verbal nouns (i.e., with the productively derived nouns) in this study:

(i) They often have a meaning denoting an action;
(ii) All of them can be found in the data thanks to their unique suffixes *-(e)ní/tí*;
(iii) Their valency is annotated according to the same guidelines in both selected corpora.

We suppose that all verbal nouns denoting an action have an obligatory Actor (ACT). Nouns denoting an abstract result of an action usually also have an Actor but it might be optional rather than obligatory as illustrated by the examples of valency frames of the noun *omezení* 'restricting/restriction' from PDT-Vallex, see (1) for denoting an action of restricting, and (2) for an abstract result of the action, i.e., restriction, with an optional Actor. Verbal nouns denoting a thing do not have an Actor in their valency frame at all, cf. two meanings of the noun *pití* 'drinking/drink' in (3) and (4).

(1) *omezení* 'restricting'
ACT(Gen,Ins,Poss) PAT(Gen,Poss) ?ORIG(*z* 'from' + Gen) ?EFF(*na* 'to' + Acc)
postupné omezení těžby.PAT *uranu ze současných 950 tun*.ORIG *na 500 tun*.EFF *ročně*
'gradual restricting of uranium mining from the current 950 tonnes to 500 tonnes per year'
(2) *omezení* 'restriction'
?ACT(Gen,Poss) ?PAT(*proti* 'on' + Dat)
omezení vlády.ACT *proti exportérům*.PAT 'restriction of the government on exporters'
(3) *pití čaje*.PAT *Petrem*.ACT 'drinking tea by Peter'
(4) *tvrdé pití* 'strong drink'

Therefore, we assume that the best way to find all verbal nouns denoting an action in our data is to search for verbal nouns that are in the data modified by an Actor (either present on the surface or added as an unexpressed but obligatory element), see (5) for the query specified in PML-TQ. Using this method, we get all the nouns denoting an action. We also get occurrences of nouns denoting an abstract result of an action in which the Actor is expressed on surface, however we believe this fact has a negligible impact on the results of our inquiry. The numbers of found verbal nouns are given in Section 6.1.

(5) An example of a query specified in PML-TQ: searching for verbal nouns denoting an action

```
t-node $a :=
[ t_lemma ~ "^.*[nt]í([_-].*)?$",
  t-node  [ functor = "ACT" ],
  a/lex.rf a-node  [ tag ~ "^N.N.*$" ] ];
  >> distinct $a
  >> give count()
```

6. Action-denoting verbal nouns in corpora of written and spoken Czech

In this section, we present the results of our search in the data of the PCEDT and PDTSC. We analyze and compare frequencies of phenomena outlined in Sections 2 and 3 which are believed to differentiate between written and spoken communication (especially frequency of verbal nouns and noun phrase complexity, see Sections 6.1 and 6.2). Exploitation of the rich and elaborate annotation scheme of the PDT corpora and use of the powerful searching tool PML-TQ enables us to particularize the distinctive features and even introduce more detailed characteristics of written and spoken communication, especially in the domain of coreference (Sections 6.3 and 6.4).

6.1. Frequency of verbal nouns and their semantic domain

Table 3 shows that while the number of occurrences of content verbs is similar in both corpora, verbal nouns are considerably more frequent in the written corpus than in the spoken one (cf. 1595 lemmas with 16283 occurrences in the PCEDT vs. only 501 lemmas with 1359 occurrences in the PDTSC). We interpret the higher number of verbal nouns denoting an action in the PCEDT as a manifestation of more condensed expression which is characteristic of written communication in general.

Given the subject matters of the texts of the used corpora (see Section 3), lexical meanings of the most frequent lemmas of verbal nouns occurring in the PCEDT and PDTSC belong to different semantic domains (see Table 3; the most frequent verbal noun that occurs in both corpora is the noun *rozhodnutí* 'decision'). Capturing personal memories and testimonies, the spoken corpus describes actions in everyday life such as exercising, meeting, having a swim, skiing, birth, travelling, and learning. The main semantic domain of the written corpus (as determined by the texts included in the corpus) is trading and related actions such as increase, reduction, taking over, making decision, negotiation, financing.

6.2. Noun phrase complexity

We carried out a quantitative analysis of combinations of participants modifying verbal nouns in both corpora. Table 4 gives relative frequencies of combinations of

Phenomenon	**Occurrences in PCEDT (written corpus)**	**Occurrences in PDTSC (spoken corpus)**
Verbal nouns denoting an action	16 283	1 359
Content verbs	99 186	102 868
Number of verbs per 1 verbal noun	6.1	75.7
Semantic domain	Trading	Activities of everyday life
The most frequent lemmas of verbal nouns	obchodování (1 323), zvýšení (590), snížení (458), převzetí (437), **rozhodnutí (357)**, jednání (325), financování (258), prohlášení (221), očekávání (190), pojištění (181), zdanění (179), omezení (167), řízení (159), obvinění (152), uzavření (147), podnikání (136), hlasování (122), získání (121), oznámení (120), zlepšení (120), snižování (113), …	cvičení (73), setkání (44), koupání (37), lyžování (33), narození (32), cestování (23), učení (21), plavání (20), vaření (17), přijímání (16), povídání (14), posezení (14), osvobození (13), vítání (12), pití (11), vyprávění (11), čtení (11), fotografování (10), psaní (10), bydlení (9), hraní (9), přání (9), tancování (9), hlídání (8),vyučení (8), stravování (8), **rozhodnutí (8)**, …
Number of lemmas of verbal nouns	1 595	501

Table 3. Frequency of verbal nouns and their semantic domain

Combinations of participants expressed on surface	**PCEDT (written corpus)** Occurrences abs.	rel.	**PDTSC (spoken corpus)** Occurrences abs.	rel.
PAT	7 003	43 %	254	18 %
ACT	1 606	10 %	101	7 %
ACT + PAT	363	2 %	7	0.5 %
PAT + ADDR	126	0.8 %	2	0.2 %
0 expr. participants	6 860	42 %	996	73 %
Other combinations	325	2 %	5	0.4 %

Table 4. Combinations of (semantic roles of) participants expressed on surface

Number of expressed participants	**PCEDT** Occurrences		**PDTSC** Occurrences	
0	6 860	42 %	996	73 %
1	8 817	54 %	359	26 %
2	585	3.6 %	10	0.7 %
3	20	0.1 %	0	0 %
4	1	0.01 %	0	0 %

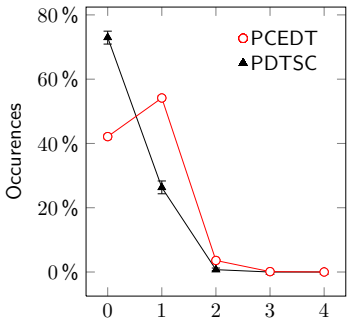


Table 5. Number of participants expressed on surface

participants expressed on the surface by any form, reflecting semantic roles of the participants. We can see that although the written corpus shows higher relative frequencies, the order of the relative frequencies of particular combinations is the same in both types of communication. The most frequent combination is the case when only Patient is expressed. The case when only Actor is expressed is the second most frequent combination, followed by the combinations Actor + Patient or Patient + Addressee, the latter of which is applicable only in the case of nouns that have an Addressee in their valency frame.[6]

Table 5 reflects the same data but focuses on the number of participants expressed on surface regardless of their semantic role. We can see that more complex noun

[6]This order of relative frequencies seems to be of general validity; for the case of verbal nouns representing five different semantic classes in the data obtained from the Prague Dependency Treebank 3.0 see (Kolářová, in press).

Phenomenon	Occurrences in PCEDT	Occurrences in PDTSC
Verbal nouns denoting an action	16 283	1 359
Chains of two verbal nouns (N_1 modified by N_2)	348 2.14 %	2 0.15 %
Chains of three verbal nouns (N_1 modified by N_2 with N_2 modified by N_3)	8	0

Table 6. Cumulation of verbal nouns

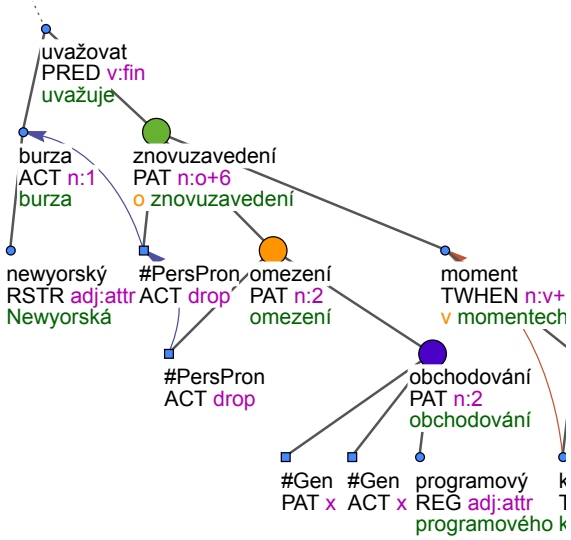
phrases are used in the written communication. The written corpus slightly prefers one expressed participant to no expressed participant, while in the spoken communication, actions are described mostly without specifying participants that take part in the situation. Combinations of two participants are rare even in the written corpus (relative frequency 3.6 % in the PCEDT and just 0.7 % in the PDTSC). Combinations of three participants appear only exceptionally.

Table 6 focuses on the case when a verbal noun is modified by another verbal noun (being a part of a prepositional phrase or in the form of prepositionless genitive). The data show that such a cumulation of verbal nouns is more frequent in the written corpus which corresponds again to the complexity of written communication. The written corpus contains 348 chains containing two verbal nouns (N_1 modified by N_2), which represents more than 2 % of all occurrences of verbal nouns denoting an action in the PCEDT. In the spoken corpus, only two such chains occur (representing just 0.15 % of all occurrences, cf. (6)). A chain containing three verbal nouns (N_1 modified by N_2 with N_2 modified by N_3) occurs only in the written corpus (8 occurrences, cf. (7) and Figure 1).

(6) *po dokončení sváření* 'after finishing welding' (PDTSC)
(7) *Nyní se obhájci UNESCO přimlouvají u prezidenta Bushe za zrušení rozhodnutí prezidenta Regana o odstoupení.* 'Now UNESCO apologists are lobbying President Bush to renege on President Reagan's decision to depart.' (PCEDT)

6.3. Degrees of explicitness

Gernsbacher (1990, p. 133—136) specifies the following scale of explicitness of anaphors (coreferring nodes): The most explicit anaphors are proper names, followed by common nouns. Pronouns are less explicit than common nouns and finally the least explicit of all referential forms are zero anaphors.



Newyorská	*burza*	*uvažuje*	*o*	*znovuzavedení*	*omezení*
New_York-NOM	*bourse-NOM*	*considers-PRS*	*about*	*reviving-LOC*	*curb-GEN*

programového	*obchodování*	*v*	*momentech,*	*kdy*	*je*	*trh*	*nestabilní.*
program-GEN	*trading-GEN*	*in*	*moments-LOC*	*when*	*is-PRS*	*market-NOM*	*volatile-NOM*

Figure 1. 'The Big Board is considering reviving a curb on program trading when the market is volatile.' (PCEDT)

We searched for participants of verbal nouns that are expressed on surface (i.e., the categories (a) and (b) in Section 4.2) and observed that among all the participants, pronouns (category (b)) are more frequent in the spoken corpus than in the written one (the relative frequency 17.0 % in the PDTSC vs. just 6.3 % in the PCEDT, see Table 7). We also checked their coreference types, that is whether their coreference is textual or grammatical. A pronoun with textual coreference is exemplified in (8) by the personal pronoun *jejich* 'their' referring to the noun *zprávy* 'messages'. A pronoun with grammatical coreference is illustrated in Figure 2 by the reflexive pronoun *svůj* 'its/their'. We suppose the Gernsbacher's scale of explicitness of anaphors can be supplemented by the opposition between pronouns with textual coreference and pronouns with grammatical coreference, the latter of which are believed to be more explicit anaphors. Our data show that pronouns with grammatical coreference (i.e., category (ba) in Section 4.2) are considerably more frequent in the written corpus than in the spoken one (cf. the relative frequency 44 % in the PCEDT and just 9.4 % in the PDTSC, see Table 7). While pronouns with textual coreference (category (bb) in Sec-

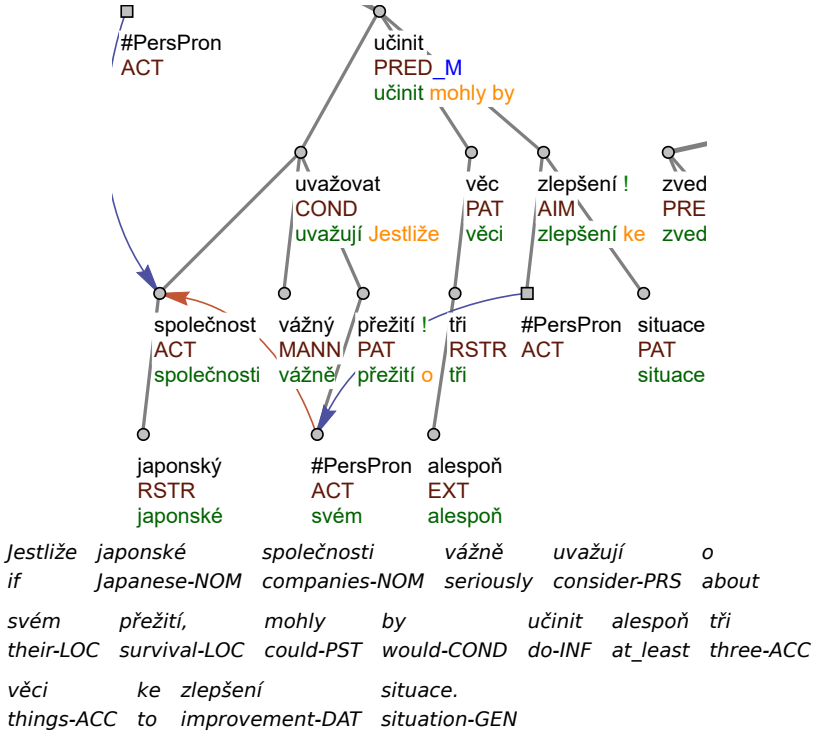


Figure 2. '*If the Japanese companies are seriously considering their survival, they could do at least three things to improve the situation.*' *(PCEDT)*

tion 4.2) account for 47.6 % in the PCEDT, they represent 81.25 % in the PDTSC. We interpret these results as a manifestation of a considerably lesser degree of explicitness in the spoken communication than in the written data.

(8) *Práce zahrnovala vybavování valutami, přijímání zpráv z těch cest a jejich předávání na jednotlivé odbory.* 'The work included providing by foreign currencies, receiving messages from the journeys and their passing on to the particular departments.' (PDTSC)

6.4. Deletions and exophoric references

In line with our expectations, there are more deletions of participants of verbal nouns in the spoken corpus than in the written one (i.e., 83.15 % in the PDTSC vs. 68 % in the PCEDT, see Table 7). Looking at the case of verbal nouns modified by no

Phenomenon	PCEDT (written corpus)	PDTSC (spoken corpus)
Participants expressed on the surface	32 %	16.85 %
Deletions of participants	68 %	83.15 %
Verbal nouns not modified by any participant from their valency frame	42 %	73 %
Participants expressed by pronouns (% of participants expressed on the surface)	6.3 %	17.0 %
– Pronouns with grammatical coreference	44 %	9.4 %
– Pronouns with textual coreference	47.6 %	81.25 %
Exophoric references (% of all participants)	0.34 %	13.37 %

Table 7. Phenomena related to degrees of explicitness

participant from their valency frame, we can see even bigger difference between the spoken and written type of communication (i.e., 73 % in the PDTSC vs. 42 % in the PCEDT, see Table 7).

Our data also reflect the difference between the context-dependent character of written language and the context-free nature of spoken communication. Adnominal participants with a context-dependent coreference refer to an antecedent in the previous context, being marked by either a textual or a grammatical coreference arrow. For example, in Figure 2, the noun *společnosti* 'companies' is referred to by a grammatical coreference arrow coming out from the node for the reflexive pronoun *svůj* 'its/their', and then this node is referred to by a textual coreference arrow coming out from the added Actor of the noun *zlepšení* 'improvement'. In contrast, adnominal participants with an extra-textual (exophoric) reference refer to something which is not mentioned in the text or speech, being indicated by a short upward arrow. For example, in the sentence illustrated by Figure 3, it is clear that children mentioned in the previous context guarded those who were at a summer camp and/or their equipment though it was not mentioned in the previous context at all; we can understand it just because we know that children usually do it when being at a camp. Adnominal participants with an exophoric reference (categories (bc) and (cc) in Section 4.2) are considerably more frequent in the spoken corpus than in the written data (13.37 % in the PDTSC vs. 0.34 % in the PCEDT).

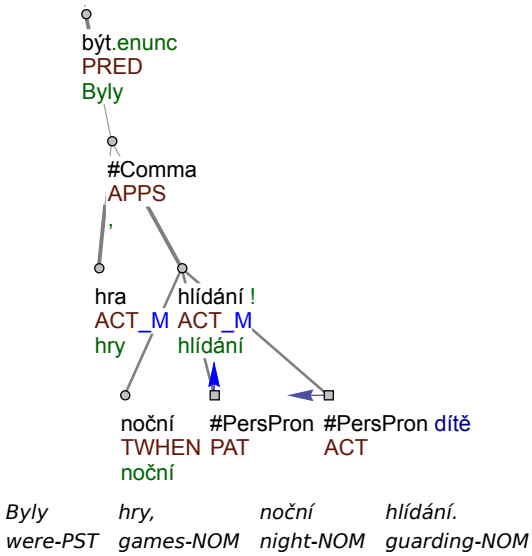


Figure 3. (To jste museli pro děti vymýšlet nějaký program? Pochopitelně. '*Did you have to think up any program for the kids? Of course.*') '*There were games, night guarding.*' *(PDTSC)*

6.5. Discussion of the results

We described differences in expressing an action by verbal nouns in the Czech part of the PCEDT (representing the written mode of communication) and in the PDTSC (representing the spoken mode of communication). Most of the results described in Sections 6.1 to Section 6.4 could not be observed without corpora with deep syntactic annotation, including annotation of deletions and coreference relations. The results confirm our hypothesis formulated in Section 2. In line with our expectations, written communication is more condensed and more complex even in the case of verbal nouns denoting an action. Although the written corpus shows higher relative frequencies, the order of the relative frequencies of particular combinations is the same in both types of communication. In the coreference domain, the complex annotation scheme enabled us to exploit the opposition between pronouns with textual and grammatical coreference (Section 6.3), demonstrating a significant difference in the degree of explicitness between written and spoken expression. The considerably higher number of deletions of participants and more frequent exophoric references give evidence about the incompleteness and context-free nature of spoken communication.

Written communication is characterized by the following features:
- Expression is more condensed, more verbal nouns per a content verb are used.
- Noun phrases are more complex which is reflected in the higher relative frequencies of combinations of participants expressed on the surface and in more frequent cumulation of verbal nouns. One expressed participant is preferred to no expressed participant.
- More explicit expressions are used (less pronouns in total, pronouns with grammatical coreference have almost the same relative frequency as pronouns with textual coreference).

In contrast, spoken communication has the following characteristics:
- Participants of verbal nouns are more often omitted on the surface and they have more often no antecedent in the context (more deletions, more extra-textual references).
- Typically no participant of a verbal noun is expressed on the surface.
- Less explicit expressions are used (approximately three times more pronouns than in written communication, considerably more pronouns with textual coreference than pronouns with grammatical coreference).

7. Conclusion

Our research into the differences between written and spoken Czech is focused on the case of verbal nouns denoting an action in the PCEDT and the PDTSC. Exploiting the annotation of valency and (co)reference relations, the results confirm our hypothesis predicting more complex noun phrases with more explicit expression in a written text and more deletions accompanied by more exophoric references in spoken communication. We support our results by numbers of occurrences of the studied phenomena in both corpora and we specify the differences between the two types of communication, providing valuable information which is hard to detect in corpora without deep syntactic annotation.

Acknowledgements

The research reported in the paper was supported by the Czech Science Foundation under the projects GA16-02196S and GA17-12624S. This work 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 second author was supported by grant no. RVO 67985840 of Czech Academy of Sciences.

Bibliography

Biber, Douglas, Stig Johansson, Geoffrey Leech, Susan Conrad, and Edward Finegan. *Longman grammar of spoken and written English*. Longman, London, 1999.

Brazil, David. *A grammar of speech*. Oxford University Press, Oxford, 1995.

Chafe, Wallace and Jane Danielewicz. Properties of spoken and written language. In Horowitz, R.; Samuels, S. J., editor, *Comprehending Oral and Written Language*. Academic Press, New York, 1987.

Čmejrková, Světla, Lucie Jílková, and Petr Kaderka. Mluvená čeština v televizních debatách: korpus DIALOG. *Slovo a slovesnost*, 65(4):243–269, 2004.

Cruse, D Alan. *Lexical semantics*. Cambridge University Press, Cambridge, UK, 1986. ISBN 0-521-27643-8.

Cvrček, Václav et al. *Mluvnice soucasné cestiny*. Karolinum, Praha, 2010. ISBN 978-80-246-1743-5.

Gernsbacher, Morton Ann. *Language comprehension as structure building*. Erlbaum, Hillsdale, 1990.

Hajič, Jan, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová, and Petr Pajas. PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation. In Nivre, J.; Hinrichs, E., editor, *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, volume 9 of *Mathematical Modeling in Physics, Engineering and Cognitive Sciences*, pages 57–68. Vaxjo University Press, Vaxjo, Sweden, 2003. ISBN 91-7636-394-5.

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 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160. European Language Resources Association, Istanbul, Turkey, 2012. ISBN 978-2-9517408-7-7.

Hajič, Jan, Eva Hajičová, Marie Mikulová, and Jiří Mírovský. Prague Dependency Treebank. In Ide, N.; Pustejovsky, J., editor, *Handbook on Linguistic Annotation*. Springer Verlag, Berlin, Germany, in press.

Halliday, Michael Alexander Kirkwood. *Intonation and grammar in British English*. Mouton, Hague, 1967.

Hausenblas, Karel. O studiu syntaxe běžně mluvených projevů. In *Otázky slovanské syntaxe: sborník brněnské syntaktické konference, 17.-21. IV. 1961*, 1962.

Heyvaert, Liesbet. *A cognitive-functional approach to nominalization in English*. Walter de Gruyter, Berlin, 2003. ISBN 3-11-017809-5.

Hoffmannová, Jana. Syntaktická stylistika mluvených projevů. In Hoffmannová, J.; Klímová, J., editor, *Čeština v pohledu synchronním a diachronním*. Karolinum, Praha, 2012.

Hoffmannová, Jana and Olga Müllerová. *Čeština v dialogu generací*. Academia, Praha, 2007.

Hoffmannová, Jana, Olga Müllerová, and Jiří Zeman. *Konverzace v češtině při rodinných a přátelských návštěvách*. Trizonia, Praha, 1999.

Hunyadi, Laszlo. Incompleteness and fragmentation in spoken language syntax and its relation to prosody and gesturing: Cognitive processes vs. Possible formal cues. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on*, pages 211–218, Budapest, Hungary, 2013. IEEE.

Kolářová, Veronika. Special valency behavior of Czech deverbal nouns. In Spevak, O., editor, *Noun Valency*, pages 19–60. John Benjamins Publishing Company, Amsterdam, The Netherlands, 2014. ISBN 9789027259233.

Kolářová, Veronika. Valence českých deverbativních substantiv reprezentujících vybrané sémantické třídy. *Prace Filologiczne*, in press. ISSN 0138-0567.

Leech, Geoffrey. Grammars of Spoken English: New Outcomes of Corpus-Oriented Research. *Language learning*, 50(4):675–724, 2000.

MacWhinney, Brian. The CHILDES project: Tools for analyzing talk. *Computational Linguistics*, 26(4):657–657, 2000.

Marcus, Mitchell, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. *Penn Treebank-3*. Linguistic Data Consortium, LDC99T42, University of Pennsylvania, 1999.

Mikulová, Marie. Annotation on the tectogrammatical level. Additions to annotation manual (with respect to PDTSC and PCEDT). Technical Report ÚFAL TR-2013-52, Prague, Czech Republic, ÚFAL MFF UK, 2014.

Mikulová, Marie and Jana Hoffmannová. *Korpusy mluvené češtiny a možnosti jejich využití pro poznání rozdílných "světů" mluvenosti a psanosti*, pages 78–92. Studie z korpusové lingvistiky. Lidové noviny, Praha, 2011. ISBN 978-80-7422-115-6.

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

Mikulová, Marie, Jan Štěpánek, and Zdeňka Urešová. Liší se mluvené a psané texty ve valenci? *Korpus – gramatika – axiologie*, 8:36–46, 2013. ISSN 1804-137X.

Mikulová, Marie, Anja Nedoluzhko, Jiří Mírovský, Jan Štěpánek, Petr Pajas, and Jan Hajič. *Prague Dependency Treebank of Spoken Czech 2.0*. Charles University, Prague Czech Republic, in press.

Müllerová, Olga. *Mluvený text a jeho syntaktická výstavba*. Academia, Praha, 1994.

Panevová, Jarmila. On verbal frames in functional generative description. Part I. *Prague Bulletin of Mathematical Linguistics*, 22:3–40, 1974.

Panevová, Jarmila. On verbal frames in functional generative description. Part II. *Prague Bulletin of Mathematical Linguistics*, 23:17–52, 1975.

Panevová, Jarmila. Ještě k teorii valence. *Slovo a slovesnost 59, č.1*, 1998.

Panevová, Jarmila. Valence a její univerzální a specifické projevy. In Hladká, Z.; Karlík, P., editor, *Čeština - univerzália a specifika. Sborník konference ve Šlapanicích u Brna 17.-18. 11. 1998*. Masarykova univerzita v Brně, Brno, 1999.

Panevová, Jarmila. K valenci substantiv (s ohledem na jejich derivaci). *Zbornik matice srpske za slavistiku*, 61:29–36, 2002.

Panevová, Jarmila. Contribution of Valency to the Analysis of Language. In Spevak, O., editor, *Noun Valency*, pages 1–18. John Benjamins Publishing Company, Amsterdam, The Netherlands, 2014. ISBN 9789027259233.

Piťha, Petr. Case frames of nouns. In Sgall, P., editor, *Contributions to functional syntax, semantics, and language comprehension,*, pages 91–99. John Benjamins, Amsterdam, Philadelphia, 1980.

Roberts, Celia and Brian Street. Spoken and written language. *The handbook of sociolinguistics*, pages 168–186, 1997.

Sagae, Kenji, Brian MacWhinney, and Alon Lavie. Adding Syntactic Annotations to Transcripts of Parent-Child Dialogs. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004*. European Language Resources Association, Lisbon, Portugal, 2004.

Schuurman, Ineke, Machteld Schouppe, Heleen Hoekstra, and Ton Van der Wouden. CGN, an annotated corpus of spoken Dutch. In Abeillé, A., S. Hansen-Schirra, and H. Uszkoreit, editors, *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 101–108. Budapest, Hungary, 2003.

Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel, Dordrecht, 1986. ISBN 90-277-1838-5.

Šonková, Jitka. *Morfologie mluvené češtiny: frekvenční analýza*. Lidové noviny, Praha, 2008.

Štěpánek, Jan and Petr Pajas. Querying Diverse Treebanks in a Uniform Way. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 1828–1835. European Language Resources Association, Valletta, Malta, 2010. ISBN 2-9517408-6-7.

Těšitelová, Marie. *Psaná a mluvená odborná čeština z kvantitativního hlediska:(v rámci věcného stylu)*. Československá akademie věd, Ústav pro jazyk český, 1983.

Urešová, Zdeňka. Building the PDT-VALLEX valency lexicon. In *Proceedings of the fifth Corpus Linguistics Conference*, pages 1–18. University of Liverpool, Liverpool, UK, 2012.

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, Prague, Czech Republic, 2015. ISBN 978-80-904571-8-8.

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

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017 39–56

Extracting Parallel Paragraphs from Common Crawl

Jakub Kúdela,[a] Irena Holubová,[a] Ondřej Bojar[b]

[a] Charles University, Faculty of Mathematics and Physics, Department of Software Engineering
[b] Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

Most of the current methods for mining parallel texts from the web assume that web pages of web sites share same structure across languages. We believe that there still exists a non-negligible amount of parallel data spread across sources not satisfying this assumption. We propose an approach based on a combination of bivec (a bilingual extension of word2vec) and locality-sensitive hashing which allows us to efficiently identify pairs of parallel segments located anywhere on pages of a given web domain, regardless their structure. We validate our method on realigning segments from a large parallel corpus. Another experiment with real-world data provided by Common Crawl Foundation confirms that our solution scales to hundreds of terabytes large set of web-crawled data.

1. Introduction

The web is as an ever-growing source of considerable amounts of parallel data that can be mined and included in the training process of machine translation systems. The task of *bilingual document alignment* can be generally stated as follows: Assume we have a set of documents written in two different languages, where a document is a plain text of any length (a sentence, a sequence of multiple sentences, or even a single word). The goal of the task is to collect all pairs of documents in different languages that are mutual translations of each other.

The majority of methods for bilingual document alignment assume that source documents are web pages and they rely on their internal structure or structure of their URLs (e.g. Resnik and Smith, 2003; Esplà-Gomis and Forcada, 2009). By this filtering for similar structure, we can lose a considerable amount of parallel data. Our proposed method is thus not based on page structure comparison. Instead, we search

 Corresponding author: bojar@ufal.mff.cuni.cz
Cite as: Jakub Kúdela, Irena Holubová, Ondřej Bojar. Extracting Parallel Paragraphs from Common Crawl. The Prague Bulletin of Mathematical Linguistics No. 107, 2017, pp. 39–56. doi: 10.1515/pralin-2017-0003.

for parallel paragraphs (or sentences) regardless their organization in the page or in the web site. By moving to these finer units, we have to rely more on the actual content of the paragraphs.

To overcome the well-known problems of text data sparseness, we use bivec (Luong et al., 2015)—a bilingual extension of currently popular word embedding model word2vec (Mikolov et al., 2013). To deal with the possibly large amount of input documents, we make use of recently studied strategies for locality-sensitive hashing (Charikar, 2002; Andoni and Indyk, 2008). Note that any finer alignment of the documents, such as sentence alignment (unless the documents consist of individual sentences), is beyond the scope of our work. However, the methods for obtaining sentence alignment for a document-aligned parallel corpus are well explored (Tiedemann, 2011) and can be easily applied to the output of our method.

Related work has been described by Roy et al. (2016) and Lohar et al. (2016). However, the discussed strategies use the original monolingual word2vec model in contrast to our solution, which makes use of the bilingual bivec model to obtain word vectors in a common vector space. Moreover, neither of the two approaches uses locality-sensitive hashing, which is utilized in our method to achieve better speed performance with regard to scalability.

The rest of the paper is structured as follows: In Section 2 we describe the proposed method. Section 3 provides the results of the experiments and Section 4 concludes and outlines future work.

2. Proposed Method

For our purposes, we refine the specification of bilingual document alignment. Let us assume we have a collection of documents in two languages of interest, organized into *bins*. Each bin holds two sets of documents, one in the source language, the other in the target language, and represents a standalone set of input documents for the original task. For each bin we want to find all the pairs of parallel documents (one in the source language, the other in the target language) present within the bin.

A bin can contain up to millions of documents and it is not required to have a balanced language distribution. Individual bins may vary in size. The binning is a way to restrict the set of considered pairs. No pairs are aligned across different bins, nor between the documents of the same language. The smaller the size of a bin, the better the quality of the resulting alignment (because fewer document pairs need to be considered). It also takes less time and memory to align a smaller bin.

When mining bilingual parallel corpora from the web, we can form a bin for every identified bilingual web domain, simply by taking all paragraphs in the two languages languages of interest, scraped from the web domain. We could be less permissive and create bins spanning several web domains, but at this moment we are leaving this option for the future.

2.1. Training Part I: Bilingual Dictionary, Bilingual Word Vectors

The proposed method is supervised and needs to be trained on an already existing sentence-aligned training parallel corpus (i.e. *seed corpus*) for the language pair we are interested in. For better clarity, we distinguish between two parts of the training process. This section describes the first part, which is depicted in Figure 1. The objective of this part is to preprocess the seed corpus and create a bilingual dictionary together with bilingual word vectors.

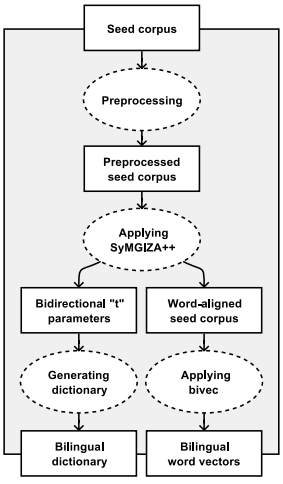


Figure 1: Method: Training part I

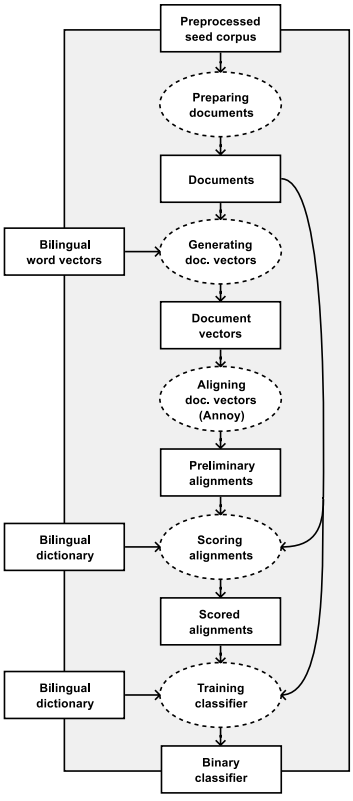


Figure 2: Method: Training part II

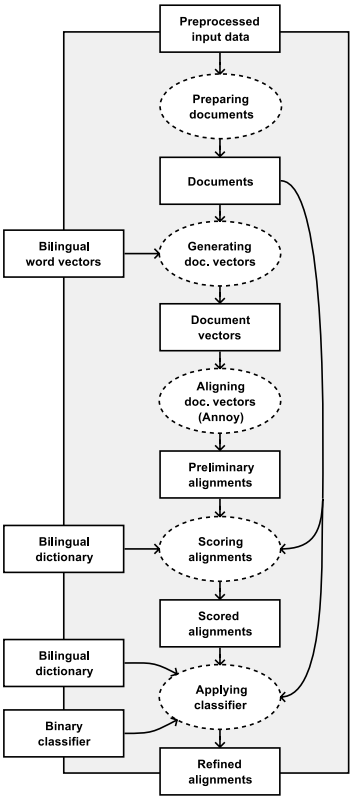


Figure 3: Method: Application

2.1.1. Preprocessing Seed Corpus

The preprocessing may involve tokenization, lemmatization, stemming, truecasing, lowercasing, removing stop words, etc. For individual language pairs, different preprocessing steps might help to gain better results. It is very important that any type of preprocessing done to the seed corpus needs to be also applied to the input data before the alignment process starts.

2.1.2. Applying SyMGIZA++

The resulting corpus from the previous step is further cleaned by removing all such pairs where one of the sentences contains more than 50 tokens or does not contain any letter from any alphabet. Then SyMGIZA++ (Junczys-Dowmunt and Szał, 2012) is executed to obtain a word alignment for the preprocessed and cleaned seed corpus. This step includes preparation of word classes and word co-occurrences which are used in the word alignment process.

SyMGIZA++ is a tool for computing symmetric word alignment models. It is an extension of MGIZA++ (Gao and Vogel, 2008), which is in turn a successor of the historically original program called GIZA++ (Och and Ney, 2003).

2.1.3. Generating Dictionary

The bilingual dictionary is built using the final IBM Model "t" parameters estimated by SyMGIZA++. Each word pair present in both directions having the harmonic mean of the "t" parameters (i.e. *weight*) over a certain threshold is included into the dictionary together with the calculated weight.

2.1.4. Applying bivec

Word embedding is a common name for a set of techniques mapping words or phrases from a vocabulary to distributed word representations in the form of vectors consisting of real numbers in a high-dimensional continuous space. Our method takes advantage of bivec—a bilingual extension of word2vec. It creates bilingual word representations when provided with a word-aligned parallel corpus.

The original word2vec is a group of models producing monolingual word embeddings. These models are implemented in form of neural networks. They are usually trained to reconstruct the contexts of words. A monolingual corpus is needed for the training. The training algorithm iterates over the words in the corpus while considering the surrounding words to be the context of the current word. One word2vec model, called continuous bag-of-words (CBOW), is trained to predict the word when given its context (without the word). Another model, named skip-gram, is trained to predict the context of a given word.

The authors of bivec proposed an extension of the original skip-gram model in the form of a joint bilingual model—bilingual skip-gram. When trained, this model can predict the context of a given word in both languages. In order to train, bivec requires a sentence-aligned parallel corpus and its word alignment. In fact, the word alignment is not strictly necessary, and if it is not provided, the system uses a simple heuristic. However, with the alignment provided, the results are better.

Our method follows the training by further processing the word-aligned seed corpus produced by SyMGIZA++ according to the recommendations of bivec. All the sequences of numbers are replaced with a unified placeholder (the symbol "`0`") and all the unknown symbols (e.g. non-printable Unicode characters) with the specially dedicated `<unk>` tag.

With the word-aligned seed corpus processed, bivec is executed to create the bilingual word vectors (i.e. embeddings) with 40 dimensions. These vectors are known to have a greater cosine similarity for context-related words, even cross-lingually. Although the number of dimensions is an unrestricted parameter, there is a reason why we keep it relatively low. The word vectors are used to calculate the aggregate document vectors with the same number of dimensions. The document vectors are then indexed using Annoy[1]—an implementation of approximate nearest neighbors search. The documentation of Annoy suggests that it works best with the number of dimensions less than 100. On the other hand, the authors of bivec conducted the tests using 40, 128, 256 and 512 dimensions. We have decided to use the only number of dimensions suitable for Annoy that has been tested with bivec.

2.2. Training Part II: Binary Classifier

The second part of the training process is illustrated in Figure 2. In this part, the method attempts to find candidate sentence pairs by realigning the preprocessed seed corpus. This is performed by employing the bilingual word vectors and locality-sensitive hashing. While working with the seed corpus, we know which of the candidate sentence pairs are correct and we exploit this knowledge to train a binary classifier. The trained classifier is then used when applying the trained method on the input data.

2.2.1. Preparing Documents

The method splits all the pairs of sentences from the seed corpus (i.e. documents) into a set of equally large bins. The last bin can be an exception. The size of a bin should be an estimate of its expected size in a real-world use case. In our experiments, we split the corpus into training bins consisting of 50,000 pairs of parallel documents, i.e. 100,000 individual documents. We believe that this amount of documents is a

[1] `https://github.com/spotify/annoy`

good upper-bound estimate of the total number of paragraphs in either of the two languages located on a typical bilingual web domain.

2.2.2. Generating Document Vectors

For each document, an associated vector is generated using the bilingual word vectors obtained in the first part of the training. To calculate the document vector, we utilize the tf-idf (i.e. term frequency - inverse document frequency) weighting scheme. For every unique document $d = (d_1, d_2, \ldots, d_n)$ a vector is generated as:

$$\text{doc\_vector}(d) = \sum_{i=1}^{n} \text{tf-idf}(d_i, d) \times \text{word\_vector}(d_i) \quad (1)$$

where $\text{tf-idf}(d_i, d)$ is a tf-idf of the term d_i in the document d and $\text{word\_vector}(d_i)$ is the bivec word vector for the term d_i. If a word vector does not exist for a given term, a zero vector is used instead.

In a smaller-scale experiment performed with our method, the tf-idf weighting scheme was compared to a plain sum of the word vectors with an equal weight. The method yielded comparatively better results using the tf-idf scheme.

2.2.3. Aligning Document Vectors (Annoy)

For each bin, independently of other bins, the following procedure is performed. A search index is built containing the vectors of all the documents in the target language. To build the search index, the method uses Annoy operating with the angular distance. Then, for every document in the source language, the index is searched to obtain k approximate nearest neighbors to its vector. This returns a list of candidate parallel documents in the target language to the document in the source language. We call them *preliminary alignments*.

Annoy is an implementation of approximate nearest neighbors search. Unlike the exact search methods, it does not guarantee to find the optimum, but in many cases it actually does. This relaxation enables it to require less resources. In particular, it needs less time, which becomes useful when dealing with larger amounts of data. Internally, Annoy uses random projections to build up a forest of search trees—an index structure for the searching process. The algorithm for building the index uses SimHash, which is a locality-sensitive hashing method.

2.2.4. Scoring Alignments

Within the preliminary alignments, the top candidates are not necessarily the optimal ones. Therefore, our method applies a scoring function to reorder the candidates

creating the *scored alignments*. This increases the probability of the optimal documents to appear higher in their candidate lists. Given the document $d = (d_1, d_2, \ldots, d_n)$ and its candidate document pair $c = (c_1, c_2, \ldots, c_m)$, where d_i and c_i are the individual terms in the respective documents, the scoring function is defined as:

$$\text{score}(d, c) = \text{length\_similarity}(d, c) \times \text{weight\_similarity}(d, c) \tag{2}$$

Both the functions length_similarity(d, c) and weight_similarity(d, c) have the range of $[0, 1]$. The idea is that the higher the result they return, the greater the possibility that the pair is parallel. The length_similarity(d, c) function compares the ratio of the documents' lengths. It is based on the probability density function of the Gaussian (normal) distribution:

$$\text{length\_similarity}(d, c) = e^{-\frac{\left(\frac{\text{length}(c)}{\text{length}(d)} - \mu\right)^2}{2\sigma^2}} \tag{3}$$

where $\frac{\text{length}(c)}{\text{length}(d)}$ is the actual ratio of the documents' lengths (i.e. total number of characters) and μ is the expected ratio with the standard deviation σ. The expected ratio with its standard deviation can be estimated using the pairs of parallel sentences from the preprocessed seed corpus. The other function weight_similarity(d, c) is based on the IBM Model 1 (Brown et al., 1993) and uses the bilingual dictionary created in the first part of the training. It is defined as:

$$\text{weight\_similarity}(d, c) = \prod_{i=1}^{n} \sum_{j=1}^{m} \frac{\text{weight}(d_i, c_j)}{m} \tag{4}$$

where weight(d_i, c_j) is the weight of the word pair $\langle d_i, c_j \rangle$ provided by the dictionary if the word pair entry exists, otherwise it equals 10^{-9} (i.e. "null weight").

2.2.5. Training Classifier

We use a binary classifier to decide whether to accept a proposed pair of documents as parallel or not. The chosen model for the classifier is a feed-forward neural network trained by back-propagating errors (Rumelhart et al., 1986). The method uses an implementation provided by PyBrain (Schaul et al., 2010). The classification is based on 4 features. All of these features have the range of $[0, 1]$. Given the document $d = (d_1, d_2, \ldots, d_n)$ and its candidate document pair $c = (c_1, c_2, \ldots, c_m)$, the following text describes all the features.

The first feature length_similarity(d, c) has been already defined in Section 2.2.4. It scores the ratio of the documents' lengths against the expected ratio. The second feature length_confidence(d, c) provides a supplementary information for the first one, which is neither reliable, nor effective when scoring pairs of short documents; however, it is substantial when comparing pairs of long documents:

$$\text{length\_confidence}(d, c) = 1 - e^{-0.01 \times \text{length}(d)} \tag{5}$$

This is a monotonically increasing function providing the model with an information of absolute length of the document d. The higher the length_confidence(d, c) is, the more authoritative the score of the length_similarity(d, c) should be deemed.

The third feature weight_similarity$_2(d_i, c_j)$ is a modified version of the one already defined (i.e. weight_similarity(d, c)). The original version was tested for the purposes of the classification, but the results were poor. The ineffectiveness could be caused by the fact that the original function lacks some proper normalization with respect to documents' sizes and returns extremely small values when comparing larger documents. The modified version is defined as follows:

$$\text{weight\_similarity}_2(d, c) = \frac{\sum_{i=1}^{n} \text{length}(d_i) \times \max_{j=1}^{m} (\text{weight}_2(d_i, c_j))}{\sum_{i=1}^{n} \text{length}(d_i) \times \text{sgn}(\max_{j=1}^{m} (\text{weight}_2(d_i, c_j)))} \tag{6}$$

where length(d_i) is the length of the term d_i and weight$_2(d_i, c_j)$ is defined as the weight of the word pair $\langle d_i, c_j \rangle$ provided by the dictionary if the entry exists; however, if the entry does not exist and the two words are identical then it equals 1, otherwise it returns 0.

Let us explain the reason for the heuristic of weight$_2(d_i, c_j) = 1$ for a pair of identical words not having an entry present in the dictionary. The same set of features is used when applying the trained method on the input data. At that moment, occurrences of new words or special terms (e.g. URLs or email addresses) are expected. The heuristic considers a pair of identical words to be a perfect translation only if the dictionary does not contain other relation.

Moreover, the weights are multiplied by the lengths of words due to an assumption that longer words are usually less frequent, carry more meaning, therefore are more important for the sentence. The definition of weight_similarity$_2(d, c)$ is an arithmetic mean of strongest relations between a source word from d and any of the target words from c, weighted by the lengths of source words. We can interpret weight_similarity$_2$ as: "Given our incomplete word translation dictionary, how likely are these two documents parallel?"

The last feature weight_confidence$_2$ supplements the third one and it can be interpreted as: "To what extent does the dictionary cover the pairs of words in these documents?". The formal definition is the following:

$$\text{weight\_confidence}_2(d, c) = \frac{\sum_{i=1}^{n} \text{length}(d_i) \times \text{sgn}(\max_{j=1}^{m} (\text{weight}_2(d_i, c_j)))}{\sum_{i=1}^{n} \text{length}(d_i)} \tag{7}$$

The process of training of the binary classifier starts by creating a supervised dataset using the scored alignments. For every document in the source language and its top candidate in the target language, a pair of input→output vectors is added into the supervised dataset as follows:

$$\begin{pmatrix} \text{length\_similarity}(d, c) \\ \text{length\_confidence}(d, c) \\ \text{weight\_similarity}_2(d, c) \\ \text{weight\_confidence}_2(d, c) \end{pmatrix} \rightarrow \begin{cases} \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if } \langle d, c \rangle \text{ are parallel} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{otherwise} \end{cases} \tag{8}$$

The input vector consists of the 4 defined features, while the output vector encodes whether the documents $\langle d, c \rangle$ are parallel or not. The first value of the output vector represents the probability of the documents to be non-parallel. The second value is complementary to the first one. Before the network is trained, the collected supervised dataset is subsampled to contain an approximately equal number of items representing parallel and non-parallel document pairs. This helps the network to be less biased by the ratio of parallel and non-parallel pairs in the supervised dataset. At this moment, it is also possible to reduce the size of the dataset to shorten the time it takes to complete the training.

For completeness, let us describe the configuration of the network. It has 4 input, 16 hidden and 2 output neurons. The hidden neurons are arranged in a single layer. The input neurons are linear, the hidden layer uses the sigmoid function and the output layer uses the softmax function.

2.3. Application

The process of applying the trained method on the input data is illustrated in Figure 3. It is almost identical with the procedure of the second part of the training (described in Section 2.2). Due to this similarity, the following text does not cover the shared parts.

The input documents have to be preprocessed in the same way as the seed corpus during the training. Then, the preprocessed documents have to be split into bins. When aligning paragraphs from the web, a bin can contain all the paragraphs for both the languages scraped from one bilingual web domain. In this scenario, the names of the web domains can be used as bin identifiers.

2.3.1. Applying Classifier

With the input dataset prepared, the process follows with the same steps as in the second part of the training. First, vectors are generated for all the documents. Then, the document vectors are aligned by searching for nearest neighbors of all the documents in the source language, resulting in preliminary alignments. In the final step, the trained classifier is used to obtain *refined alignments*. For every document in the source language and its top candidate in the target language the trained network is activated in the same way it has been trained. The second value of the output vector represents the confidence that the two documents are parallel. If the confidence is greater than a user-defined threshold, the document pair ends up in the resulting refined alignments (i.e. *extracted corpus*).

3. Experiments

Our experiments are solely focused on the Czech–English language pair. The first experiment is carried out using CzEng 1.0 (Bojar et al., 2012)—a Czech–English sentence-aligned parallel corpus. By realigning the CzEng 1.0 corpus, we can evaluate the quality of the results automatically. The second experiment uses more realistic and noisy data provided by Common Crawl Foundation.[2] The organization produces and maintains an open repository of web-crawled data that is universally accessible and analyzable.

3.1. Prealigned Data (CzEng 1.0) Experiment

The corpus consists of all the training sections (packs 00–97) of CzEng 1.0 in the plain text, untokenized format. It includes 14,833,358 pairs of parallel sentences collected from various domains (e.g. fiction, legislation, movie subtitles, parallel web pages, etc.). By default, the pairs are shuffled, meaning they are not grouped by their domains. The shuffled corpus is split exactly in half into a *head* (i.e. seed corpus) for training and a *tail* for evaluation. The whole procedure is illustrated in Figure 4.

The preprocessing step consists of tokenization and lowercasing. The head is cleaned by excluding all such pairs where one of the sentences contains too many tokens or does not contain any letter from any alphabet. The tail is cleaned by only

[2] http://commoncrawl.org/

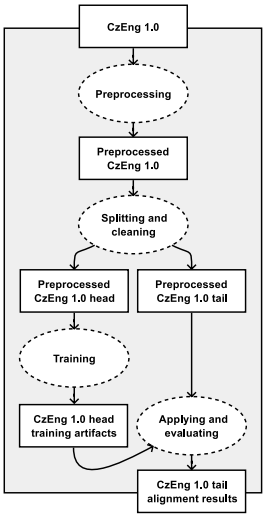


Figure 4: Prealigned Data (CzEng 1.0) Experiment

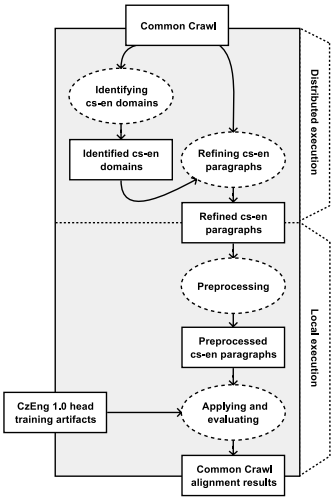


Figure 5: Web Data (Common Crawl) Experiment

applying the latter of the two mentioned criteria. The pairs containing overly long sentences are removed from the head to get better quality word alignment.

During the training, SyMGIZA++ uses the union method for the final symmetrization of the word alignments, while bivec uses the bilingual skip-gram model and runs for 10 iterations. The pairs of parallel sentences from the head are distributed into a set of artificial bins to form a dataset for the training of the classifier. Each bin contains 50,000 pairs of parallel documents from various domains, i.e. 100,000 individual documents. As discussed above, we believe this amount of documents to be a good upper-bound estimate of the total number of paragraphs in either of the two languages (Czech and English) located on a typical Czech–English web domain. Annoy builds search indexes with 500 trees and during each search it inspects up to 20,000 nodes to return 20 candidates. The classifier is trained using approximately 20% of all the available pairs of Czech documents with their corresponding top English candidates. Additionally, the supervised dataset contains nearly as many parallel pairs as non-parallel. The classifier's network is trained for 20 epochs with 1% learning rate.

The trained method is used to realign the tail. The pairs of parallel sentences from the tail are distributed into artificial bins in the same manner as those from the head. In this scenario, each bin simulates a web domain with 50,000 Czech and 50,000 English paragraphs that we want to align. In contrast to real-world websites, these are

perfectly parallel, meaning that all the content is available in both languages. The original alignment of the tail is forgotten and it does not affect the evaluation process is any way. The confidence threshold of the classifier is set to 50%. The refined alignments represent a subset of all the pairs of Czech documents with their top English candidates that the classifier accepts to be parallel.

In the preliminary alignments of the tail, 50.30% of all the top candidates are exact matches. However, this ratio is more satisfactory in the scored alignments—71.30%. This means that the scoring of the preliminary alignments is an important step in the whole process. The overall effectiveness of our method (after the application of the binary classifier), when realigning the tail part of CzEng 1.0, is listed in Table 1. Of all the existing pairs of parallel sentences 63.02% were detected, and 93.74% of all the detected pairs were correct.

Recall (%)	63.02
Precision (%)	93.74

Table 1: Prealigned Data (CzEng 1.0); Experiment: Effectiveness

The computer used for the execution has Intel® Xeon® CPU E5-2630 v3 (20 MB Cache, 2.40 GHz) and 128 GB of memory. Table 2 lists the approximate time durations of the individual steps of the experiment.

3.2. Web Data (Common Crawl) Experiment

The second experiment deals with the non-parallel, real-word, noisy data acquired from the web. The language pair of our interest is again Czech–English. The procedure uses the training artifacts created in the first experiment, namely the dictionary, bilingual word vectors and the trained classifier.

The input data are obtained from the July 2015 dataset provided by Common Crawl Foundation and consist of approximately 1.84 billions of crawled web pages, taking about 149 TB of disk space in an uncompressed format. To store and process this large volume of data we use Hadoop—an HDFS (Shvachko et al., 2010) cluster and the MapReduce (Dean and Ghemawat, 2004) framework.

The procedure of the experiment is illustrated in Figure 5. It starts with a distributed execution running two MapReduce jobs. The first job creates a list of web domains containing at least some Czech and English paragraphs, i.e. contents of <p> HTML tags. Parsing of the HTML is done using jsoup[3] and language detection is per-

[3] https://jsoup.org/

Activity	Duration (hh:mm)
Preprocessing	
Tokenizing and lowercasing	00:08
Splitting and cleaning	00:05
Training part I	
Applying SyMGIZA++	13:21
Generating dictionary	00:10
Applying bivec	01:01
Training part II	
Generating document vectors	00:37
Aligning document vectors (Annoy)	05:52
Scoring alignments	02:49
Training binary classifier	01:29
Application	
Generating document vectors	00:45
Aligning document vectors (Annoy)	07:04
Scoring alignments	04:10
Applying binary classifier	00:47

Table 2: Prealigned Data (CzEng 1.0); Experiment: Time Duration

formed by language-detector.[4] Due to the unsatisfactory effectiveness of language-detector on shorter texts, paragraphs having less than 100 characters are discarded. In the future, the parsing could be modified to consider also other HTML tags.

The list of web domains is further filtered to keep only those having the ratio of Czech to English paragraphs within the interval $(0.01, 100)$. This filtering discards all domains with very unbalanced language distribution. The output of the first job contains $8,750$ identified bilingual domains.

The second MapReduce job extracts the Czech and English paragraphs for all the identified bilingual web domains. In order to provide the second job with the file containing the accepted domains, Hadoop Distributed Cache is utilized. The output of the second job contains $5,931,091$ paragraphs for both languages, namely $801,116$ Czech and $5,129,975$ English. These paragraphs are aligned with our method in a local execution and the results are evaluated. All the settings remain the same as in the first experiment, except the threshold of the classifier is changed to 99%. The precision is favored over the recall. The extracted paragraph-aligned parallel corpus contains

[4] https://github.com/optimaize/language-detector

114,771 pairs from 2,178 domains, having in total 7,235,908 Czech and 8,369,870 English tokens. Table 3 lists the most frequent web domains contributing to the extracted corpus. The size of the extracted corpus is comparable with the amount of Czech–English parallel data acquired by the related project focused on mining the Common Crawl datasets (Smith et al., 2013).

The quality of the extracted corpus is evaluated manually on a set of 500 randomly selected paragraph pairs. The inspected pairs are categorized into the categories displayed in Table 4. A pair of paragraphs is considered to be a human translation if it seems like created by a human. If the translation of the pair seems cumbersome, it is labeled as a product of machine translation. A partial match represents a situation, when one paragraph is incomplete regarding the content of the other one. Everything else is labeled as a mismatch. If we consider the pairs belonging to the two categories of human and machine translation as true positives, then the estimation of precision is 94.60%.

Source Domain	**Paragraph Pairs**	**Ratio (%)**
europa.eu	23457	20.45
eur-lex.europa.eu	15037	13.11
windows.microsoft.com	11905	10.38
www.europarl.europa.eu	8560	7.46
www.project-syndicate.org	2210	1.93
www.debian.org	2191	1.91
support.office.com	1908	1.66
www.esa.int	1308	1.14
www.eea.europa.eu	1299	1.13
www.muni.cz	1206	1.05
⋮	⋮	⋮
Total	114,711	100.00

Table 3: Web Data (Common Crawl); Experiment: Web Domains

Manual evaluation of the method's recall is complicated and therefore it is performed using only one selected web domain—`www.csa.cz`, the official website of Czech Airlines. The input dataset contains 68 Czech and 87 English paragraphs for this domain. These paragraphs are manually aligned, creating the *desirable alignments*. When evaluated, the desirable alignments contain 44 paragraph pairs, of which 42 also appear in the corpus extracted by our method. Additionally, the extracted corpus include 1 extra pair subjectively regarded as a mismatch. Table 5 shows the effectiveness of our method evaluated for the `www.csa.cz` web domain.

Category	Count	Ratio (%)
Human translation	466	93.20
Machine translation	7	1.40
Partial match	13	2.60
Mismatch	14	2.80
Total	500	100.00

Table 4: Web Data (Common Crawl); Experiment: Evaluation (500 Paragraph Pairs)

Recall (%)	95.45
Precision (%)	97.67

Table 5: Web Data (Common Crawl)
Experiment: Effectiveness (`www.csa.cz`)

The Hadoop cluster used for the distributed execution of the two MapReduce jobs consists of 3 management nodes and 24 worker nodes. The management nodes run components like front-end, HDFS NameNode and MapReduce History Server. Every node of the configuration has Intel® Xeon® CPU E5-2630 v3 (20 MB Cache, 2.40 GHz) and 128 GB of memory. The total disk space available on the cluster is 1.02 PB. The HDFS operates with a replication factor of 4. The rest of the procedure, i.e. the local execution, is done on one node of the cluster. Table 6 contains the approximate time durations of the individual steps of the experiment.

4. Conclusions and Future Work

The majority of methods for bilingual document alignment search for pairs of parallel web pages by comparing the similarity of their HTML structures. Our method does not depend on any kind of page structure comparison. We are able to efficiently identify pairs of parallel segments (i.e. paragraphs) located anywhere on the pages of a web domain, regardless of their structure.

To verify the idea of our method, we have performed two experiments focused on the Czech–English language pair with both prealigned and real-world data. These experiments show satisfactory results, implying that the proposed method is a promising baseline for acquiring parallel corpora from the web.

Nevertheless, there is still some room for improvement. First of all, our method does not consider word order at any stage during the aligning process. The scoring function and the features of the classifier could be extended to take word order into ac-

Activity	Duration (hh:mm)
MapReduce framework	
Identifying cs-en domains	11:58
Refining cs-en paragraphs	11:38
Local Execution	
Tokenization and lowercasing	00:09
Generating document vectors	00:58
Aligning document vectors (Annoy)	01:13
Scoring alignments	03:42
Applying classifier	00:39

Table 6: Web Data (Common Crawl); Experiment: Time Duration

count. Then, there is an asymmetric nature of our method, meaning that it yields different results if the source and the target languages are swapped. The method could perform the alignment for both directions and the results could be symmetrized. This might help to achieve an even higher precision.

Another possibility would be to extend our method with some kind of structural comparison, for instance, in form of a new feature for the classifier, that would compare the structural origin of the input documents (e.g. XPath of <p> tags, in case of aligning paragraphs from the web).

Finally, we have used our method only in a single-node environment so far. This is largely because we have worked with relatively small sets of documents (not more than $15,000,000$). However, the method is designed to be able to run in distributed fashion. Bins with input documents represent independent and isolable tasks. With the method trained in a local execution, these tasks could be distributed across multiple nodes of a cluster. This could increase the throughput of our method, and hence decrease the overall execution time.

Acknowledgements

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 644402 (HimL). This work has been using language resources stored and distributed by the LINDAT/CLARIN

project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Bibliography

Andoni, Alexandr and Piotr Indyk. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM*, 51(1):117–122, Jan. 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327494. URL http://doi.acm.org/10.1145/1327452.1327494.

Bojar, Ondřej, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

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

Charikar, Moses S. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM. ISBN 1-58113-495-9. doi: 10.1145/509907.509965. URL http://doi.acm.org/10.1145/509907.509965.

Dean, Jeffrey and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251254.1251264.

Esplà-Gomis, Miquel and Mikel L. Forcada. Bitextor, a free/open-source software to harvest translation memories from multilingual websites. In *Proceedings of the workshop Beyond Translation Memories: New Tools for Translators MT*, Ottawa, Ontario, Canada, August 2009.

Gao, Qin and Stephan Vogel. Parallel Implementations of Word Alignment Tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-10-7. URL http://dl.acm.org/citation.cfm?id=1622110.1622119.

Junczys-Dowmunt, Marcin and Arkadiusz Szał. SyMGiza++: Symmetrized Word Alignment Models for Machine Translation. In Bouvry, Pascal, Mieczyslaw A. Klopotek, Franck Leprévost, Malgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybinski, editors, *Security and Intelligent Information Systems (SIIS)*, volume 7053 of *Lecture Notes in Computer Science*, pages 379–390, Warsaw, Poland, 2012. Springer. URL http://emjotde.github.io/publications/pdf/mjd2011siis.pdf.

Lohar, Pintu, Debasis Ganguly, Haithem Afli, Andy Way, and Gareth J.F. Jones. FaDA: Fast Document Aligner using Word Embedding. *The Prague Bulletin of Mathematical Linguistics*, 106(1):169–179, Oct. 2016. ISSN 0891-2017. doi: 10.1515/pralin-2016-0016. URL https://doi.org/10.1515/pralin-2016-0016.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. Bilingual Word Representations with Monolingual Quality in Mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States, 2015.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist.*, 29(1):19–51, Mar. 2003. ISSN 0891-2017. doi: 10.1162/089120103321337421. URL http://dx.doi.org/10.1162/089120103321337421.

Resnik, Philip and Noah A. Smith. The Web As a Parallel Corpus. *Comput. Linguist.*, 29(3): 349–380, Sept. 2003. ISSN 0891-2017. doi: 10.1162/089120103322711578. URL http://dx.doi.org/10.1162/089120103322711578.

Roy, Dwaipayan, Debasis Ganguly, Mandar Mitra, and Gareth J. F. Jones. Representing Documents and Queries as Sets of Word Embedded Vectors for Information Retrieval. *CoRR*, abs/1606.07869, 2016. URL http://arxiv.org/abs/1606.07869.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038%2F323533a0.

Schaul, Tom, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

Shvachko, Konstantin, Hairong Kuang, Sanjay Radia, and Robert Chansler. The Hadoop Distributed File System. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-1-4244-7152-2. doi: 10.1109/MSST.2010.5496972. URL http://dx.doi.org/10.1109/MSST.2010.5496972.

Smith, Jason R., Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt Cheap Web-Scale Parallel Text from the Common Crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1383, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-1135.

Tiedemann, J. *Bitext Alignment*. Synthesis digital library of engineering and computer science. Morgan & Claypool, 2011. ISBN 9781608455102. URL https://books.google.cz/books?id=IMCIGSMB5k0C.

Address for correspondence:
Ondřej Bojar
bojar@ufal.mff.cuni.cz
ÚFAL MFF UK (Linguistics)
Malostranské náměstí 25
11800 Praha
Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 2017 57–66

Česílko Goes Open-source

Jernej Vičič,[a b] Vladislav Kuboň,[c] Petr Homola[d]

[a] University of Primorska, The Andrej Marušič Institute
[b] Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute
[c] Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University
[d] Semantek Ltd.

Abstract

The Machine Translation system Česílko has been developed as an answer to a growing need of translation and localization from one source language to many target languages. The system belongs to the shallow parse, shallow transfer RBMT paradigm and it is designed primarily for translation of related languages. The paper presents the architecture, the development design and the basic installation instructions of the translation system.

1. Introduction

The system Česílko (language data and software tools) was first developed as an answer to a growing need of translation and localization from one source language to many target languages. The starting system belonged to the Shallow Parse, Shallow Transfer Rule-Based Machine Translation – (RBMT) paradigm and it was designed primarily for translation of related languages. The latest implementation of the system uses a stochastic ranker; so technically it belongs to the hybrid machine translation paradigm, using stochastic methods combined with the traditional Shallow Transfer RBMT methods. The source code that is now published as open-source under the MIT license (The MIT License (n.d.), 2016) is almost the same as that which is presented in (Homola and Kuboň, 2008) with some slight modifications made to compile the code on GNU/Linux.

This article presents the architecture, the development design and the basic installation instructions of the translation system and is organised as follows. The state

 Corresponding author: jernej.vicic@upr.si
Cite as: Jernej Vičič, Vladislav Kuboň, Petr Homola. Česílko Goes Open-source. The Prague Bulletin of Mathematical Linguistics No. 107, 2017, pp. 57–66. doi: 10.1515/pralin-2017-0004.

of the art is presented in Section 2, followed by the description of the history of the translation system in Section 3. Section 4 presents the outline of software architecture, followed by the description of the installation process in Section 5. The article concludes with a list of tested environments in Section 6 and a discussion and further work in Section 7.

2. State of the Art

The framework presented in this paper can be attributed to the paradigm of Fully Automatic Machine Translation (FAMT), which comprises every automatic translation of natural languages with no user intervention ("EAMT", 2010). More specifically, the framework focuses on the translation of related languages, one of the most suitable paradigms for this domain is the Shallow Transfer Rule-Based Machine Translation. It has a long tradition and has been successfully used in a number of MT systems, some of which are listed in Section 2.1. Shallow-transfer systems usually use a relatively linear and straightforward architecture, where the analysis of a source language is usually limited to the morphemic level.

The latest version of Česílko uses a stochastic ranker, so technically it is a hybrid machine translation system framework, using stochastic methods combined with the traditional Shallow Transfer RBMT.

2.1. Existing MT Systems for Related Languages

A number of experiments in the domain of machine translation for related languages have led to the construction of more or less functional translation systems. The systems are ordered alphabetically:

- Altinas (Altintas and Cicekli, 2002) for Turkic languages.
- Apertium (Corbi-Bellot et al., 2005) for Romance languages.
- Dyvik, Bick and Ahrenberg (Dyvik, 1995; Bick and Nygaard, 2007; Ahrenberg and Holmqvist, 2004) for Scandinavian languages.
- Česílko (Hajič et al., 2000a), for Slavic languages with rich inflectional morphology, mostly language pairs with Czech language as a source.
- Ruslan (Oliva, 1989) full-fledged transfer based RBMT system from Czech to Russian.
- Scannell (Scannell, 2006) for Gaelic languages; between Irish (Gaeilge) and Scottish Gaelic (G'aidhlig).
- Tyers (Tyers et al., 2009) for the North Sámi to Lule Sámi language pair.
- Guat (Vičič et al., 2016) for Slavic languages with rich inflectional morphology, mostly language pairs with Slovenian language.

3. The history of the MT system Česílko

The idea to develop an MT system for very closely related languages has actually been inspired by the request of the company SAP to support the localization of their products into Slavic languages. The original idea was relatively simple – the texts were supposed to be translated by human translators from the original languages (English and German) into Czech and then automatically translated into a number of related languages. The translations served as a support for human translators from the original languages into the target Slavic languages. The automatically translated texts were added to the translation memories, from which they were retrieved only in the event that no better translations already existed in the translation memory (this can easily be achieved by setting a penalty for machine translated texts in the translation memory). The details of this setup can be found for example in (Hajič et al., 2000a).

The actual architecture of the system called Česílko has been developed between the years 1998 and 2000, it was for the first time described in (Hajič et al., 2000b), more detailed description can be found in (Hajič et al., 2000a).

Figure 1 shows the architecture of the most popular translation system for related languages Apertium (Corbi-Bellot et al., 2005) and its predecessor, Česílko (Hajič et al., 2003), designed primarily for the translation between Slavic languages.

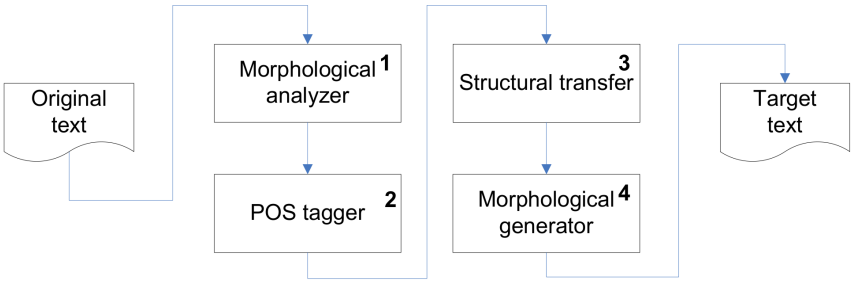


Figure 1. The modules of a typical Shallow Transfer RBMT system, this architecture was adopted in the first version of Česílko.

.

The system exploited the work from the previous MT system RUSLAN cf. (Oliva, 1989) and (Hajič, 1987), also aiming at the translation between related Slavic languages. Its development started in mid eighties, the system aimed at the automatic translation of texts from a limited domain (manuals of operating systems of main-

frame computers). The system RUSLAN system was designed as a traditional transfer-based system with full morphological and syntactic analysis of Czech as a source language and the syntactic and morphological synthesis of Russian as the target language. The system was designed with the assumption that the close relatedness of both languages would primarily be reflected in the transfer phase and in the dictionary of the system. This assumption turned out to be wrong.

The transfer phase, originally very simple, had to be substantially enlarged in the process of testing the system. The simplicity of the transfer phase was achieved also due to the fact that the lexical transfer was not handled by an independent transfer module it was actually performed in the dictionary lookup phase, and the transfer module actually covered only structural transfer and primarily dealt with syntactic differences only. The subtle syntactic differences between Czech and Russian were still differences after all, and as such they had to be handled by specific transfer rules. The number of those specific transfer rules grew together with the amount of the text used for testing the system.

The lessons learned in this project clearly indicated that the strategy chosen for RUSLAN did not exploit the similarity of both languages to the desired extent and that the closeness of both related languages actually did not have a major positive effect on the quality of the results achieved. It was negatively influenced by the complexity of the system and the close relatedness of languages actually called for much simpler architecture.

Instead of the morphological similarity, the simplified architecture of the Česílko exploited the syntactic similarity of the related languages and also chose a more similar target language – Slovak. The syntactic analysis of the source language (Czech) was completely removed due to the assumption that both languages have in fact identical syntax (the existing differences being only marginal). A stochastic morphological tagger performed the disambiguating role of the syntactic analysis. It took the ambiguous information provided the morphological analysis module of Czech and provided a single morphological tag with the highest probability in the given context. The translation module then translated both the lemma provided by the morphological analysis and the tag (the target language morphology uses slightly different tagset and therefore the translation of the source language tag was necessary). This information was then exploited by the morphological synthesis module of the target language. No syntactic information was used for the synthesis of the target language, the system strongly relied on the syntactic similarity of both languages.

The results of the Czech-to-Slovak translations were good enough to justify further experiments (the translated text required less than 10% post-editing operations in order to obtain a high quality translation.). The next two target languages added were Polish and Lithuanian. The results of these experiments have been described in (Hajič et al., 2003). The experiments clearly showed that the most important phenomenon, which makes the automatic translation of related languages easier, is their syntactic similarity. From the lexical point of view, Lithuanian is much less similar to Czech

than it is to Polish. Lithuanian also belongs to a different language family (Baltic languages), while Czech and Polish are both Western Slavic languages. On the other hand, the syntactic differences between Czech and Polish or Czech and Lithuanian are of a similar nature, and thus the fact that the translation results of these two language pairs are of a similar quality strongly supports the hypothesis that the syntactic similarity is the decisive factor in the MT of closely related languages.

Several other experiments with other target languages (e.g., Lower Sorbian, Macedonian, Russian etc.) have been performed in subsequent years. All these experiments, described for example in (Homola and Kuboň, 2004) and (Dvořák et al., 2006), showed that the more similar is the syntax of the source and the target language, the better are the translation results. It became clear that the better translation quality can be possible by changing the architecture through hybridization of the original architecture by the involvement of a stochastic ranker instead of the tagger. This substantial improvement of the architecture has been described both in the Ph.D. thesis of Petr Homola and in several articles such as (Homola and Kuboň, 2008) and (Homola and Kuboň, 2010). The change of the architecture actually improved the translation quality for all target languages, but, unfortunately, for less related ones, the improvement was only relatively small.

4. The architecture

The Česílko system has a very simple architecture. It exploits the close similarity of both languages at all linguistic levels. There is no full-fledged analysis of the source text, the system adopts a simplistic approach of ignoring syntactic differences and focusing on morphology and lexica. A partial (shallow) parser is implemented mainly to cope with possible high degree of morphological ambiguity present in a morphologically rich languages such as Czech or Slovenian. Figure 2 shows the architecture of the latest version of Česílko that is being published as open-source.

The translation system is organized as a pipeline of four programs each using the output of the preceding program. The morphological analyzer *morph* searches for all possible applications of the surface forms in the source morphological dictionary. The output of this module is fed to the shallow syntactical analyzer *syntan* implemented as a bottom-up chart parser. The formalism of Q-systems has turned out to suit the requirements although there are already plans to change the setting. The *transfer* module searches for the source – target lemma pairs in the bilingual dictionary, applies the changes in charts and later uses the target morphological dictionary to prepare the paths in the chart in the target language. The traversal of all possible paths through the chart gives a set of translation candidates. The output of the *transfer* ranked by a target language model – *ranker*, which is a simple trigram language model although the architecture allows a transparent change of the latest element. *Ranker* selects the best translation from the list of candidates according to the language model. The shallow parser produces highly ambiguous results because it is generally impossible to

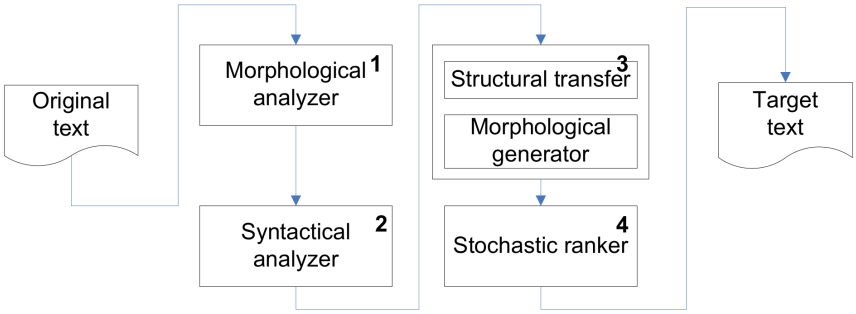


Figure 2. The modules of new Česílko system.

.

fully morphologically disambiguate the input sentences on the basis of local context only. The task of selecting the best result is left till the end of the processing chain, to a stochastic ranker of generated target language sentences. A simple trigram-based language model (trained on word forms without any morphological annotation) sorts out ″wrong″ target sentences. An extended description of the architecture and all modules can be found in (Homola and Kuboň, 2008).

4.1. Česílko strengths

One of the weaknesses of the shallow-transfer RBMT system is how they deal with the ambiguities introduced by the morphological analysis. The ambiguities can be eliminated with a set of rules using in a form of CG grammar (Karlsson et al., 1995) or using statistical POS taggers such as (Brants, 2000). Such architectures are presented in Figure 1. The errors introduced at the early phases of the translation pipeline have a big effect on the translation quality. Eliminating the modules leads to an explosion of the number of translation candidates (morphologically rich languages produces millions of candidates) (Vičič et al., 2009).

4.2. Translation quality evaluation

The evaluation of the translation quality was done previously in (Homola and Kuboň, 2008) and (Homola and Kuboň, 2010) on language data that is not part of the open-source distribution. The best results were obtained with the translation pair Czech – Slovak (only this direction), the results using the HTER (Snover et al., 2006) metric were 3.15 %.

5. Installation process

The latest version of Česílko was coded for OS X in Objective C, the reason for this was solely pragmatic (the main developer was using OS X). This experiment was mainly focused on making Česílko available as open-source and on porting Česílko to GNU/Linux and Windows, through Cygwin (Steinhauser, 2013) or MinGW (Shpigor, 2013) (Minimalist GNU for Windows). The port is based on GNUstep library/runtime (Chisnall, 2012). The GNU/Linux port comprised on adjusting the libraries and parts of the source code to accustom small changes in code (mainly just changing header files).

The source code and a test dataset is available on GitHub.[1] The test dataset supports the language pair Czech – Slovak, it is a small subset of the data used in (Homola and Kuboň, 2008). Following tools and libraries need to be installed on a fresh installation of Ubuntu in order to successfully compile and start Česílko:

- *git* – fast, scalable, distributed revision control system,
- *clang* – C, C++ and Objective-C compiler (LLVM based),
- *GNUstep Development Environment* – development tools,
- the latest version of the libobjC2 from GNUstep (not available in repository).

Following tools and libraries need to be installed on a fresh installation of mac OS in order to successfully compile and start Česílko:

- *Xcode* – Xcode is an integrated development environment (IDE).

A quick cheat-sheet of the installation on the Ubuntu operating system is presented in Figure 3. A script that installs the development environment (and many other things) enables easy install.

When all parts of the development environment are prepared, simply go to the code directory and start the make process: *make; make install*. A test translation pipeline is prepared in the Makefile. Start test target by typing: *make test*; a successful installation will present a translation of the test example (in Czech) into the Slovak language.

6. Tested environments

The code was successfully compiled and started (used) on these platforms:

- latest LTS editions of Ubuntu (Ubuntu 16.04 and 14.04). It was compiled with Ubuntu clang version 3.8.02ubuntu4 (tags/RELEASE_380/final).
- latest editions of the OS X El capitan and macOS Sierra. It was compiled with Apple LLVM version 7.0.2 (clang700.1.81).

[1] GitHub: `https://github.com/cesilko/cesilko`

```
sudo apt-get install git
# Copy an Objective-C installation script from
# http://wiki.gnustep.org/index.php/GNUstep_under_Ubuntu_Linux
# and start the script using root privileges.
git clone https://github.com/cesilko/cesilko
cd cesilko
make
make test
```

Figure 3. The installation steps for the build environment and Česílko. In the last command, the test target shows a typical usage with the toy dataset. For the OS X, install xcode and ignore the first three lines.

.

7. Discussion and further work

The paper has presented the history of the Shallow Parse/Transfer RBMT system Česílko which was transformed to a hybrid MT paradigm with the change in the architecture by the addition of a stochastic ranker. The system has been made available to the research community by open-sourcing the source code under MIT license (The MIT License (n.d.), 2016). At the moment the supported environment is GNU/Linux (tested on Ubuntu 16.04 amd 14.04 platform), although the code was developed on MacOS and compiles well on that operating system. The Windows platform is supported only using Cygwin or MinGW, so this is one of the first steps that need to be performed in the near future.

While the architecture of Česílko is really simple, it is modular and flexible so one can easily add new modules. One possible addition is a fully-fledged parser based on unification and a broad-coverage valency lexicon, which would allow for more distant language pairs. Another module being worked on is pragmatic interpretation of the source text, particularly the translation-by-abduction approach (Hobbs and Kameyama, 1990), which is planned for future versions instead of the statistical ranker to evaluate translation candidates on logical grounds.

Bibliography

Ahrenberg, Lars and Maria Holmqvist. Back to the Future? The Case for English-Swedish Direct Machine Translation. In *Proceedings of The Conference on Recent Advances in Scandinavian Machine Translation*. University of Uppsala, 2004.

Altintas, Kemal and Ilyas Cicekli. A Machine Translation System between a Pair of Closely Related Languages. In *Proceedings of the 17th International Symposium on Computer and Information Sciences (ISCIS 2002)*, page 5. CRC Press, 2002.

Bick, Eckhard and Lars Nygaard. Using Danish as a CG Interlingua: A Wide-Coverage Norwegian-English Machine Translation System. In *Proceedings of NODALIDA, Tartu*. University of Tartu, 2007.

Brants, Thorsten. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference*, pages 224–231. Seattle, WA, 2000.

Chisnall, David. A New Objective-C Runtime: From Research to Production. *Queue*, 10(7): 20:20—-20:24, 2012. ISSN 1542-7730. doi: 10.1145/2330087.2331170. URL `http://doi.acm.org/10.1145/2330087.2331170`.

Corbi-Bellot, Antonio M, Mikel L Forcada, and Sergio Ortiz-Rojas. An open-source shallow-transfer machine translation engine for the Romance languages of Spain. In *Proceedings of the EAMT conference*, pages 79–86. HITEC e.V., 2005.

Dvořák, Boštjan, Petr Homola, and Vladislav Kuboň. Exploiting Similarity in the {MT} into a Minority Language. In *Proceedings of the 5th {SALTMIL} Workshop on Minority Languages*, pages 59–64, Paris, France, 2006. European Language Resources Association. ISBN 2-9517408-2-4.

Dyvik, Helge. Exploiting Structural Similarities in Machine Translation. *Computers and Humanities*, 28:225–245, 1995.

"EAMT". European Association for Machine Translation, 2010. URL `http://www.eamt.org/`.

Hajič, Jan. RUSLAN: an MT System Between Closely Related Languages. In *Proceedings of the third conference of the European Chapter of the Association for Computational Linguistics*, pages 113–117. Association for Computational Linguistics, 1987.

Hajič, Jan, Jan Hric, and Vladislav Kuboň. Česílko – an MT system for closely related languages. In *Proceedings of ACL 2000*, pages 7–8, 2000a.

Hajič, Jan, Jan Hric, and Vladislav Kuboň. Machine translation of very close languages. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 7–12. Association for Computational Linguistics, 2000b.

Hajič, Jan, Petr Homola, and Vladislav Kuboň. A simple multilingual machine translation system. In Hovy, Eduard and Elliott Macklovitch, editors, *Proceedings of the MT Summit IX*, pages 157–164, New Orleans, USA, 2003. AMTA.

Hobbs, Jerry R and Megumi Kameyama. Translation by Abduction. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*, COLING '90, pages 155–161, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics. ISBN 952-90-2028-7. doi: 10.3115/991146.991174. URL `http://dx.doi.org/10.3115/991146.991174`.

Homola, Petr and Vladislav Kuboň. A translation model for languages of acceding countries. In *Proceedings of the {EAMT} Workshop*, 2004.

Homola, Petr and Vladislav Kuboň. A method of hybrid MT for related languages. In *Proceedings of the IIS*, pages 269–278. Academic Publishing House EXIT, 2008.

Homola, Petr and Vladislav Kuboň. A Method of Hybrid MT for Related Languages. *Control and Cybernetics*, 39(2):421–438, 2010. ISSN 0324-8569.

Karlsson, Fred, Atro Voutilainen, Juha Heikkila, and Arto Anttila, editors. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, 1995. ISBN 3110141795. URL `http://portal.acm.org/citation.cfm?id=546590`.

Oliva, Karel. *A Parser for Czech Implemented in Systems Q*. MFF UK Prague, 1989.

Scannell, Kevin P. Machine translation for closely related language pairs. In *Proceedings of the Workshop Strategies for developing machine translation for minority languages*, pages 103–109. Genoa, Italy, 2006.

Shpigor, Ilya. *Instant MinGW Starter*. Packt Publishing, 2013.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231. Citeseer, AMTA, 2006. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.4369&rep=rep1&type=pdf`.

Steinhauser, Martin Oliver. Installation guide to Cygwin. In *Computer Simulation in Physics and Engineering*, chapter Appendix B, pages 445–447. Walter de Gruyter GmbH & Co. KG, 2013.

The MIT License (n.d.). The MIT License, 2016. URL `http://www.opensource.org/licenses/MIT`.

Tyers, Francis M, Linda Wiechetek, and Trond Trosterud. Developing prototypes for machine translation between two Sámi languages. In *Proceedings of EAMT*. HITEC e.V., 2009.

Vičič, Jernej, Petr Homola, and Vladislav Kuboň. A method to restrict the blow-up of hypotheses of a non-disambiguated shallow machine translation system. In *Proceedings of the RANLP*, pages 460–464, Borovec, Bulgaria, 2009. Association for Computational Linguistics.

Vičič, Jernej, Petr Homola, and Vladislav Kuboň. Automated implementation process of machine translation system for related languages. *Computing & Informatics*, 35(2):441 – 469, 2016.

Address for correspondence:
Jernej Vičič
`jernej.vicic@upr.si`
University of Primorska, UP IAM
Muzejski trg 2, 6000 Koper, Slovenia

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 107 APRIL 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`.