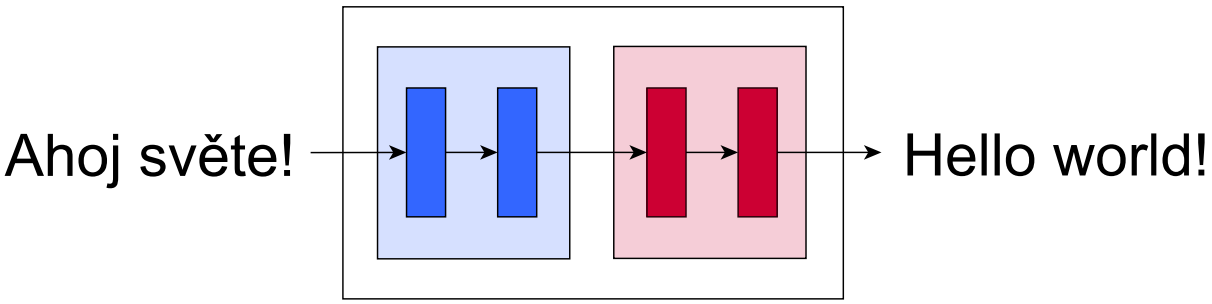
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Machine Translation Using Artificial Neural Networks

Jonáš Holcner*



Abstract
The aim of this work is to develop a neural machine translation system (NMT). This is a machine translation system based on neural networks. Specifically, the system is based on an encoder-decoder architecture, created with recurrent neural networks enabling sequence-to-sequence translation. The system is built with Keras and Tensorflow libraries and is tested against Moses statistical machine translation tool. This work does not bring any specific new state-of-the-art model but shows an insight into the topic. The result is an open source python library provided for public use.

Keywords: neural machine translation — NMT — recurrent neural networks — RNN — LSTM — encoder-decoder architecture — sequence to sequence — seq2seq — keras — moses — BLEU

Supplementary Material: Downloadable Library

*xholcn01@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

In the recent years, there is a significant increase in usage of a machine learning and an artificial intelligence. This is because, the capacity and performance of computers caught up with the available amount of data that is being produced every day, as to build and train large enough machine learning models. Nowadays, neural networks are widely capable of recognizing images, transcribing spoken language and most interestingly for this paper, they are quite capable in translating sequences from a one language to another.

The biggest advantage of the modern neural machine translation (*NMT*) approach is that it does not have some of the problems the traditional machine translation systems had. Instead of translating only parts of sentences or phrases, NMT has the ability to learn and translate directly whole sequences in end-to-end fashion.

The goal of this work is to develop and try out such system and provide an out of the box usable library. The solution proposed in this paper makes use of the encoder-decoder architecture.

The result is python package *nmt* built with Keras and Tensorflow machine learning libraries. Experiments with test datasets were conducted using this package, evaluated with the standard BLEU score. Results were compared with the system produced by the Moses [1] statistical machine tool.

2. Previous Works

The first idea of the recurrent neural networks (RNN) comes from the nineties [2]. The vanilla RNN, however, had a problem with long term dependencies because of the vanishing and exploding gradients [3].

Thus came improved variants of the RNN – long short term memory (LSTM) [4, 5] and its simpler version, gated recurrent unit (GRU) [6]. These units have a memory, that stores and changes information in it over time, enabling the network to remember long term dependencies.

Works [7, 8, 9] showed that then state-of-the-art language models were built using recurrent neural networks. This layed a foundation for the neural machine translation as language models are the vital part. The advantage of neural language model is that it learns embeddings (Figure 1) in a continuous space for the words, which provides the model with more context it can learn from.

$$V = [\text{farmer, cow, steak, eats, } \dots]$$

$$oneHot_{steak} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

$$wordEmbedding_{steak} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{300} \end{bmatrix}$$

Figure 1. Difference between common representation and word embeddings representation of words from vocabulary V. Common one-hot sparse vector represent the word with 1 on the position in the vector that corresponds with the index of the word in the vocabulary. Word embedding dense vector on the other hand represents the word with a for example vector of size 300, which holds some semantic value of the word and thus can be better utilized by the neural network.

Different variants of learning the word embbeddings are shown here [10, 11, 12, 13]. Pre-trained word embeddings, for example on some very large data set, can be used to boost the performance of a NMT system, which would have to otherwise learn those embeddings by itself.

The encoder-decoder architecture was proposed in [14] and was used for rescoring hypotheses produced by a phrase-based system with a successful improvement. Sutskever et al. [15] then showed how to use the encoder-decoder architecture for direct sequence-to-sequence translation and comes with the best results at the time. Furthermore, it was found that reversing order of the words in all source sentences (reverse encoder), improves models performance, by introducing short term dependencies between the source and the target sentence.

Upon this builds the work [16] which shows even better results with the bi-directional encoder which is an encoder that process sequences in both forward and reverse order. What is even more important, they address the problem of the encoder-decoder approach, where the meaning of the translated sentence is captured in a fixed-length vector and that can be problematic for translating long sentences. The proposed remedy is so called *attention* mechanism which lets the model, at the time of decoding, look at the most important words from the source sentence for the currently translated word, resulting in even better performance.

As translation is an open-vocabulary problem, the NMT systems have to somehow handle the words not present at the time of the training. This was typically done by using out-of-vocabulary tokens and by using very large vocabularies, which causes the models to be very memory and performance demanding. A solution for this can be sub-word units [17, 18], that are shown to be more efficient, help with rare and unknown words and improve the results.

Current state-of-the-art results [19, 20], uses all of the techniques described, showing that they can be successfully applied on large production data sets.

Johnson et al. [21] shows, that with no changes to the model architecture, one model can be used to learn to translate between many languages, even to produce translations between languages that it was not explicitly trained on (zero-shot translations).

3. Seq2seq translation with encoder-decoder architecture

Sequence-to-sequence (**seq2seq**) means that the process of translating takes one whole sentence at the input, in a source language, and its meaning translates into a sequence in a target language. This can be better in contrast to other methods like word-based translation or phrase-based translation as in seq2seq the whole meaning of the source sequence is translated at once and can keep the precise meaning. Seq2seq is modeling probability of a target sentence based on a source sentence as shown in Figure 2.

The neural network architecture, used for this kind of modeling, is **encoder-decoder**. Both the encoder

and the decoder are recurrent neural network (Figure 3), usually one or more layers of LSTM or GRU.

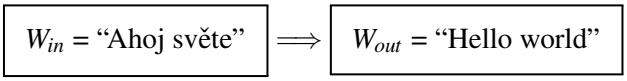


Figure 2. Seq2seq models a probability $P(W_{out}|W_{in})$. Meaning that the model learns to predict a sentence W_{out} based on a sentence W_{in}.

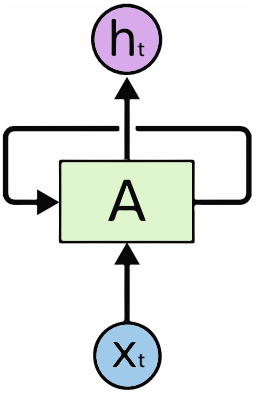


Figure 3. RNN are neural networks with loops. They process sequence of data and an output h_t from each time step t is fed as an input to the time step $t+1$. The image is taken from [22].

The encoder takes embeddings of the input tokens (words or subwords) and processes the input sequence in a source language into a fixed-length vector, the so called "thought" vector – it captures the essence of the given sentence.

The hidden state of a decoder is initialized with this output from an encoder. The decoding process is started with a start token, which is given to the decoder as the first input. The decoder then generates a sequence in the target language until it reaches an end token that tells it to stop. There is a difference between the time when model is being trained and between the time when it is used to predict sequences. In the training time, the decoder is fed the correct, expected output. This is called "teacher forcing" [23]. In the inference time, the decoder is fed its own output the each time step. The Figure 4 shows the encoder-decoder architecture.

Equations of the encoder-decoder architecture:

$$m_t^{(f)} = M_{f_t}^{(f)} \tag{1}$$

$$h_t^f = \begin{cases} RNN^{(f)}(m_t^{(f)}, h_{t-1}^{(f)}) & \text{if t} \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$$m_t^{(e)} = M_{e_{t-1}}^{(e)} \tag{3}$$

$$h_t^e = \begin{cases} RNN^{(e)}(m_t^{(e)}, h_{t-1}^{(e)}) & \text{if t} \geq 1, \\ h_{|F|}^f & \text{otherwise.} \end{cases} \tag{4}$$

$$p_t^{(e)} = \text{softmax}(W_{hs} h_t^{(e)} + b_s) \tag{5}$$

- $RNN^{(f)}$ - encoder, $RNN^{(e)}$ - decoder
- $M_{f_t}^{(f)}$ - embedding for encoder input in time t
- $M_{e_t}^{(e)}$ - embedding for decoder input in time t
- $h_t^{(f)}$ - encoder hidden state in time t
- $h_t^{(e)}$ - decoder hidden state in time t
- W_{hs} - weights
- b_s - bias

An embedding is looked up for every word in time t (Eq. 1). Then the hidden state of the encoder is computed (Eq. 2). After processing of the whole input sequence, there should be enough context stored for the decoder initialisation. Then, as similar to the encoder, an embedding is looked up for the decoder input (Eq. 3). Only now the word is from the time $t-1$ as the decoder generates a new word based on the last one. In the time t_0 the decoder is fed the starting symbol $\langle s \rangle$. Eq. 4 computes the hidden state of the decoder and shows that in the time t_0, the decoder is initialised with the encoder final state. The final output probability is computed with the $softmax$ function (Eq. 5).

An in depth overview of NMT is given in [24].

4. Implementation of the NMT system

The presented system is implemented in Python with Keras library using Tensorflow backend. The result is a python package *nmt*, published on github[1].

4.1 Data preprocessing

Some of the Moses scripts are used for preprocessing of the data. To tokenize sequences, *tokenizer.perl* is used. To select proper uppercase first letter for each token, *truecase.perl* is used. Cleaning is done with *clean-corpus-n.perl* so that only sequences with max length of 15 tokens are kept in the dataset. To split tokens into sub-word units, *subword-nmt*[2] scripts are used.

[1] https://github.com/jojkos/neural-machine-translation

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

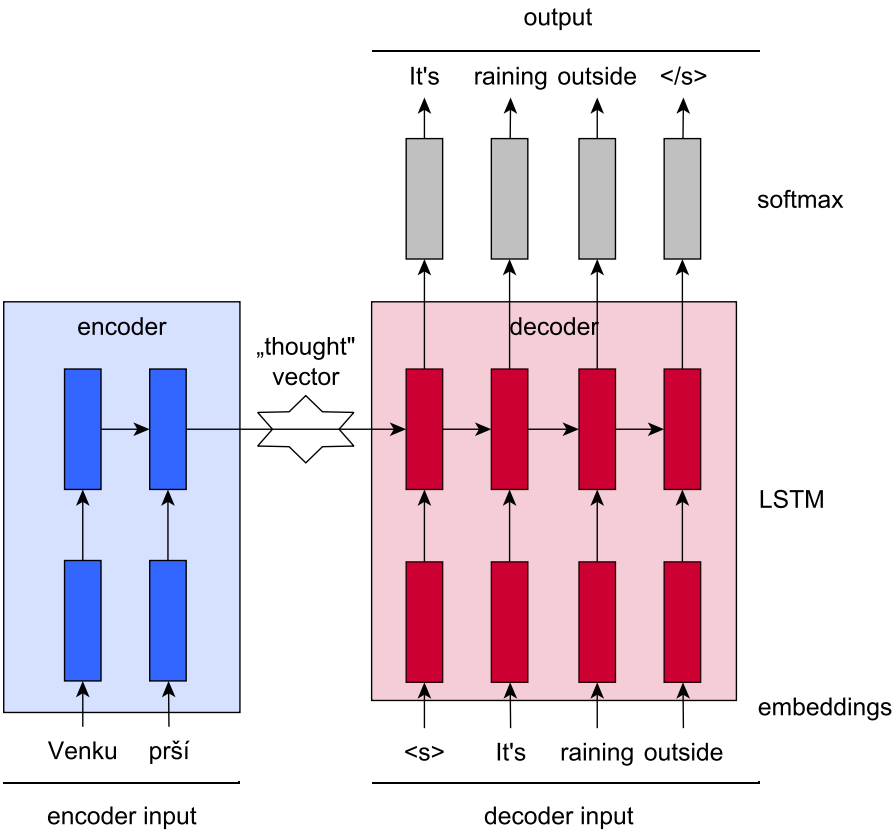


Figure 4. The encoder-decoder architecture shown on translation from a Czech sentence to its English equivalent. The encoder processes embeddings of the input sequence and produces the fixed-length "thought" vector. This vector is used as initial state of the decoder, it tells it from what context should it produce the output in the target language. The prediction is started with the $\langle s \rangle$ starting token. Then the decoder is fed either correct output tokens during the training time or its own output, from the time $t-1$, during inference time, until it generates the ending $\langle /s \rangle$ token.

4.2 Implementation details

Size of a source and a target vocabulary is clipped to a maximum size chosen in an experiment and unknown words are replaced with $\langle unk \rangle$ token. The package can be used with pre-trained word embeddings in *fastText*[13] format. To enhance performance, bidirectional encoder is implemented, it is possible to use multiple layers for both the decoder and the encoder and during training, dropout can be specified and teacher forcing is used. LSTM is used as an RNN unit in both the encoder and the decoder. During translation generation, beam search is used to find best hypothesis with maximum score.

5. Experiments and results

This sections presents results of experiments conducted with the *nmt* package compared with results made with baseline system created with the Moses tool. The evaluation is done with the standard BLEU metric. BLEU ([25]) is an algorithm for automatic evaluation of translation systems. It compares translated sentences with a reference translations. Those scores are averaged over the whole dataset and the output is a number between 0–100, indicating how similar is the candidate text to the reference texts, the higher the more similar.

Datasets are taken from the yearly conference on machine translation (*WMT* – formerly **w**orkshop on statistical **m**achine **t**ranslation) conference. In particular, datasets from WMT17[3] are used. They contain text data from different domains like subtitles, books, news and websites. Size of each dataset is shown in Table 2.

For all experiments, pre-trained word embeddings with size of 300 dimensions were used, obtained from facebookresearch[4].

Experiments ran on VUT FIT computational clus-

[3] http://data.statmt.org/wmt17/translation-task/preprocessed/

[4] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

ter[5] on GPU. RMSprop was used as optimizer with learning rate 0.001. Each experiment ran until the result of loss function for validation dataset did not decrease for 5 epochs.

Table 1. Examples of translations.

original sentences
1. budoucnost se jevila černě.
2. někdo mu musí dát lekci.
3. nejdůležitější výsledky studie jsou uvedeny v tabulce 4 a na obrázku 3.
4. uvidíme, jak se ta schůze bude odvíjet.
5. Evropská unie je druhým největším obchodním partnerem ASEAN.
6. 13. prosinec
reference translation
1. the future was grey.
2. somebody's got to teach him a lesson.
3. the key results of the trial are listed in Table 4 and Figure 3.
4. see how the meeting goes.
5. the European Union is ASEAN' s second largest trading partner.
6. December 13
baseline Moses system translation
1. the jevila black.
2. someone must give him a lesson.
3. the study results are provided in Table 4 and the picture 3.
4. we'll see how the meeting is odvíjet.
5. the European Union's second largest partner ASEAN.
6. 13 December
presented system translation
1. the future was red.
2. someone needs to give him a lesson.
3. the most important results of the two and two are available in the level of the 3.
4. let's see how the meeting goes.
5. the European Agreement is the most attractive partner of the European Union.
6. 13.

First, several experiments with different hyperparameters were conducted on language pair Cs→En, to find out optimal hyperparameters for the system. The same training dataset was used to train baseline system with Moses tool. Best performing NMT model was with hyperparameters: 2 layers LSTM of size 1000 layers for encoder, 2 LSTM of size 1000 for decoder, 15000 maximum vocabulary size, dropout of 0.1, and beam size 15. Best performing model and baseline system were evaluated with test dataset. Results are in the Table 3 and examples of translations are in the Table 1.

[5] http://www.fit.vutbr.cz/CVT/cluster/

Table 2. Number of sentence pairs for different used datasets.

dataset	sentence pairs
training $Cs \rightarrow En$	1000000
validation $Cs \rightarrow En$	2000
test $Cs \rightarrow En$	2000
training $Cs \rightarrow En + En \rightarrow De$	2000000
test $En \rightarrow De$	2000
test $Cs \rightarrow De$	2000

Table 3. Resulting BLEU score on the $Cs \rightarrow En$ test dataset of the baseline system and of the best performing model created with *nmt* package. The system made with Moses tool is showing better performance.

system	BLEU score
baseline	23.08
created	9.87

Another experiment was conducted to find out how the performance of the model will be affected when its trained to translate on more language pairs. Training dataset included $Cs \rightarrow En$ training dataset used in previous experiments, plus another 1000000 sentence pairs for $En \rightarrow De$ pair. Model was then tested with test datasets for $Cs \rightarrow En$ pair, $En \rightarrow De$ pair and for $Cs \rightarrow De$ to try out zero-shot translation on a language pair that the model was not explicitly trained on. Results are shown in Table 4.

Results shows that models created with the published system are capable of learning to translate between two languages and even between multiple language pairs with at least some precision. Farther investigation is needed to find out why the performance does not reach the performance of the current state-of-the-art systems and of the baseline system.

Table 4. Resulting BLEU score for different test datasets, each for one language pair. Translation was done by one model trained on multiple language pairs. Results shows, that the model is now capable to translate $Cs \rightarrow En$ and $En \rightarrow De$ for the price of having worse results for the former. Zero-shot translation for $Cs \rightarrow De$ does not show much success.

language pair	BLEU score
$Cs \rightarrow En$	7.10
$En \rightarrow De$	7.39
$Cs \rightarrow De$	0.14

6. Conclusions

The goal of this paper was to explore neural machine translation and to develop a translation system with the encoder-decoder architecture. This system was built, published on github and tested against Moses statistical translation tool.

The best BLEU score on $Cs \rightarrow En$ translation of the created system is **9.87** while the system built with Moses has score **23.08**. When the system was trained on multiple languages, score for $Cs \rightarrow En$ pair dropped to **7.10** but the same model was able to translate even for $En \rightarrow De$ pair with score **7.39**. Experiment with zero-shot was not very successful and would probably require bigger training dataset. Although presented system does not give excellent results, it shows that the core is working.

Next plan is to find the reason of the subpar results and improve them. Another step would be to add attention module.

Acknowledgements

I would like to thank my supervisor Ing. Igor Szőke, Ph.D. for his guidance.

References

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

[2] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[5] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, October 2000.

[6] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[8] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[9] Tomáš Mikolov. *Statistické jazykové modely založené na neuronových sítích*. PhD thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2012.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[11] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, 2013.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word

vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[17] Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocký. Subword language modeling with neural networks. In *Subword Language Modeling with Neural Networks*, 2011.

[18] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015.

[19] Yonghui Wu et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[21] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016.

[22] Christopher Olah. Understanding lstm networks, 2015. [Online; navštíveno 3.12.2017].

[23] Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. pages 4601–4609. 2016.

[24] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619, 2017.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.