English-Dutch Machine Translation in the Europarl Dataset

Project 2.6 / Natural Language Processing

OVIDIU VICTOR TĂTAR and JACQUELINE SZEIBERT

Machine translation is a task in natural language processing, where a model translates a sequence of text from one language to another.
In this report we experiment with neural machine translation models on the Europarl dataset [8], encompassing over 15 years from the *Verbatim Report of Proceedings* of the European Parliament.[1] The Dutch-English corpus includes parliamentary debates from 1996 til 2011, which have been professionally translated to both languages.

First we present some insights we gathered about the dataset in section 1 and explain our preprocessing steps in section 2. Then we present our fundamental model architecture in section 3 and experiment with different variations of said architecture in section 4 presenting our final results in subsection 4.2. Finally we conclude this report in section 5.

1 DATA ANALYSIS

The Europarl website reports that the Dutch-English dataset contains

- 1 997 775 paired translations,
- 50 602 994 Dutch words and
- 49 469 373 English words.

We arrive at slightly lower numbers (1 960 122 translation pairs) after our preprocessing step described in the next section, but the preprocessed dataset still contains a large amount of professionally translated texts.

First we noticed that there is some markup, indicating language the current parliamentary representative was originally speaking. We decide to visualize these in Figure 1, where we can clearly see that the distributions vary greatly in the Dutch and English versions. Therefore these markup language codes represent noise that we necessarily have to remove later on, as they degrade the quality of training- and test data. Interestingly, the most frequently spoken languages according to this markup is English, followed by German, French and Polish.

We also decide to look into which contemporary years are mentioned in the parliamentary debates: In Figure 2 we see that the distributions match up much better. Excitingly the European Parliament seems to have planned ahead: Despite the data only including proceedings up to 2011, goals for 2020 were frequently discussed as can be clearly seen in the graph.

Next, we investigate some basic linguistic properties of the correspondences. In the Figure 3 and Figure 4 the distribution of the top 15 most frequent characters is depicted, which may be relevant to any character-based translation models developed on the dataset. While the vocal “e” clearly plays a big role in both languages, in Dutch it is actually significantly more frequent than any other alphabetic character. Overall, both distributions contain similar characters, however their rank and frequency differs greatly.

Finally, we analyse the most frequent words in both languages, depicted in Figure 5 and Figure 6. As both texts talk about the same contents we would expect these to approximately match up. And indeed: “de” means “the” in Dutch! “van” means “of”, “en” means “and”, “zijn” means “are”. The

[1]The dataset is available online at https://www.statmt.org/europarl/.

Authors’ address: Ovidiu Victor Tătar, o.tatar@tu-berlin.de; Jacqueline Szeibert, j.szeibert@tu-berlin.de.

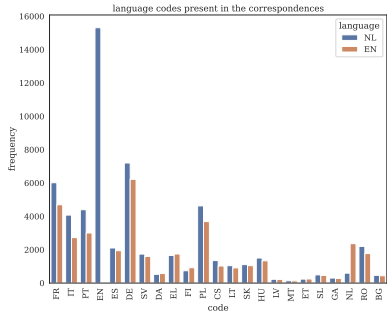

Fig. 1. occurrences of language markup in both the English and Dutch version

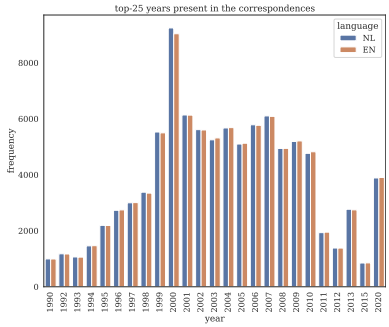

Fig. 2. occurrences of contemporary year numbers in both the English and Dutch version

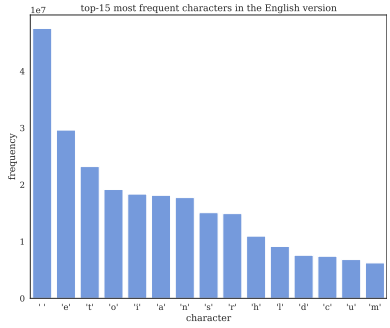

Fig. 3. most frequent English characters

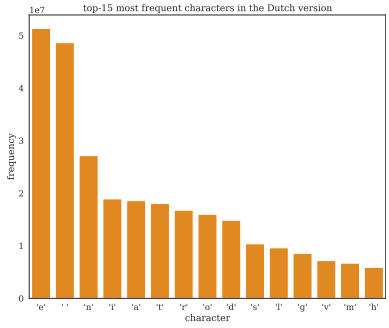

Fig. 4. most frequent Dutch characters

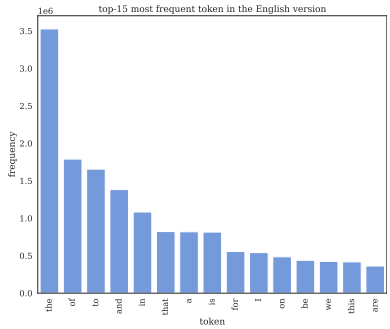

Fig. 5. most frequent English words

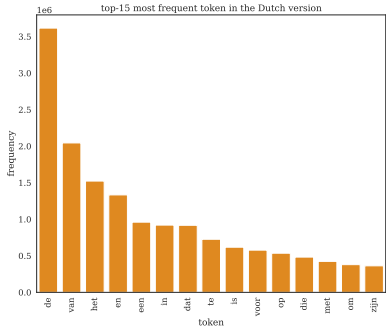

Fig. 6. most frequent Dutch words

rank of the frequency differs slightly, but these words roughly match up. As was to be expected, the most frequent words would be considered *stop words* by most NLP-researchers.

2 PREPROCESSING

The Europarl dataset contains a complete translation of a text in a given language (e.g. Dutch) to English. The text and translation are split into multiple parts that should match up, such that each line from the files form a pair of corresponding translations. Unfortunately, the alignment of the

different parts is not perfect, resulting in many parts without matching translation. We find that 1.717% of the data (34304 translation pairs) is affected by fixable misalignments.

As a first step we perform an realignment of translation pairs; As such texts with missing translations are merged with surrounding translation-pairs, forming a correct example again.

Noise removal. Through manual analysis we identified some sources of noise in the dataset which we deal with using the following rules:

- remove markup such as enumerations and codes identifying the original language of the segment such as "DE", "EN", "FR", ...
- remove trailing punctuation
- remove trailing or leading non-word characters

Normalization.

- remove extra whitespace
- normalize manually identified specific unicode characters to their corresponding ASCII-equivalents
- perform an approximate transliteration for Cyrillic and Greek characters, as many of them look identical to characters from the Latin alphabet and have thus mistakenly been used in English and Dutch correspondences alike

We do not lowercase the text, as we believe that capitalization can convey important semantic information.

The dataset is already in Unicode Normalization Form C, so we do not perform further Unicode normalization. In particular we do not apply *Compatibility*-based normalization forms, as these exchange characters for other visually similar, but semantically distinct ones.[2]

2.1 Tokenization

We train a unigram SentencePiece tokenizer for each language we apply our models to [9]. SentencePiece is an algorithm to create subword-level tokenizers via unsupervised learning, which means the resulting tokenizers can split words into multiple sub-tokens and therefore reliably process entire text corpora despite using a fixed-size vocabulary. Further still, previously unseen words can be represented as a sequence of multiple sub-tokens, meaning that a model using a SentencePiece tokenizer may function well even when presented with novel words. Only when presented with a unknown character will the tokenizer insert unknown tokens (<unk>). For simplicity we use a vocabulary size of 16000, despite the different linguistic characteristics of English and Dutch.

For encoder-decoder models, tokenizers are usually required to also insert special tokens: *beginning of sequence* tokens (<s>) and *end of sequence* tokens (</s>). The primary purpose of these is to give a model's decoder an input when predicting the first token (<s>) and the last token (</s>) [14].[3]

3 MODEL ARCHITECTURE

We develop and train our model using Keras [4] / KerasNLP [16], a Python framework around TensorFlow [5] focused on developing neural networks.

We build a encoder-decoder sequence-to-sequence model as pioneered by [14]. The architecture we use is visualized in Figure 7: The encoder receives one-hot encoded tokens which it transforms to a dense representation using an embedding layer. We then use one long short-term memory (LSTM)-layer to encode the sequence of embeddings and provide the resulting states to the decoder.

[2]See https://www.unicode.org/reports/tr15/ for further information on Unicode normalization.
[3]See also https://huggingface.co/blog/encoder-decoder#background for a more in-depth explanation.

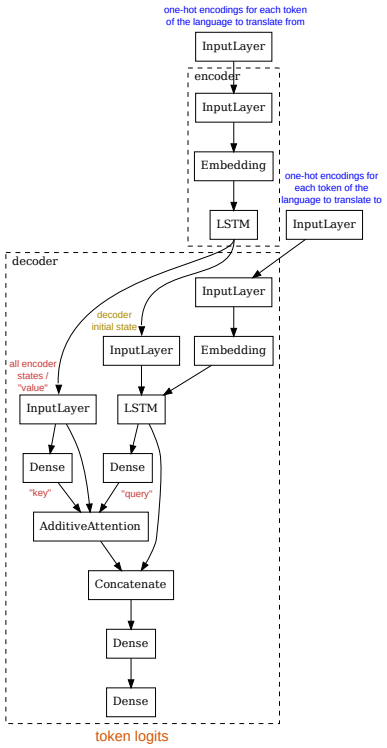

Fig. 7. Our fundamental model architecture; nodes connected only to the attention-layer are removed on model-variants without the attention layer.

The decoder embeds a target sequence of tokens and produces a sequence of decoder-states. Target tokens represent the target translation, preceded by a <s>; This prevents the decoder to simply copy the target translation, as it does not know which token will follow at each step. At the same time, the model will know in the next step which tokens *it should have predicted* previously. This encoder-decoder mechanism is described in greater detail in [14]. The optional attention decoder-layer described in greater detail in the subsequent segment makes our architecture akin to [1].

The decoder's final output layer contains the so-called *logits* for each output token. A decoder-state is expanded to a dimensionality of the vocabulary size (16k in our case), thus creating a distribution over each token. By applying a softmax function one would get a probability distribution over the next most probable tokens. However we defer the computation of probabilities to the loss-function and text generation stage, as this may provide better numerical stability.[4]

Attention. We also experiment with using an attention mechanism: Attention is a weighted sum of value vectors, where the weights are determined by a compatibility function between a vector of queries and a vector of keys [15].

We use Bahdanau-style additive attention, making our architecture akin to that of [1]. This style of architecture achieved good results in the past.[5]

[4]See: https://datascience.stackexchange.com/a/73182

[5]The best performing model until 2016 in the WTM2014 benchmark is such an attention-based encoder-decoder RNN, see: https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-german

If the attention-layer is enabled, we concatenate the resulting attention-weights to the decoder-states from the LSTM before passing them to the logits-layer.

Text Generation. To generate text we employ a so-called *beam-search*, which, in the context of sequence-to-sequence models, was first pioneered by [6].

The beam-search iterates over the model multiple times until it reaches a previously configured maximum length or an </s> token, which the model is trained to use when it is done with the current translation. Starting from an initially empty translation-result of <s>, the beam search provides the model with the text-segment to be translated and the text generated so far, and appends the next most probable token to the generated translation. A number of k "beams" is kept, to be able to keep the top-k generated texts and not only follow the path of highest probability. This is done, because the most probable path may not result in the best translation, as sequences with higher probability could be hidden behind less probable predictions.[6]

4 EXPERIMENTS

We randomly sample 8% (156 809 examples) of the dataset as training data for our models. For the test data randomly select 0.05% (981 examples) from the remaining dataset (excluding the training data). While this does not seem like much data to judge the performance by, to generate a single translation the beam-search can take several seconds, so it is not feasible for us to use more data.

We first conduct multiple preliminary trials, testing out different model variants, and present our final results afterwards in subsection 4.2.

Metrics. We evaluate using the BLEU [11] and METEOR [2] metrics as made available by the NLTK python-library [3]. Both metrics range from 0 (low quality translation / no overlap between reference and model generated translation) and 100 (perfect match). A perfect score of 100 on every generated translation is usually hard to achieve and not necessarily desirable, because it may indicate that the model is overfitted.

As these metrics require tokenized inputs, we use NLTK's pretrained Punkt-tokenizer [7] to tokenize texts during evaluation. The tokenizer has been trained on English texts only, but we believe tokenization rules to be similar for the European languages we study. We also perform stemming on the resulting tokens using the well-known Porter-Stemmer [12], respectively the appropriate version for each language from its continuation in the Snowball project[7]. For BLEU we use ORANGE-smoothing [10], which corresponds to add-one smoothing. It should be noted that METEOR uses WordNet to account for synonyms in the translation; as such its scores may show higher quality in the evaluation of English translations.

4.1 Preliminary Trials

We configure the beam-search to output a maximum of 16 tokens and use 2 search-beams. This means that a plurality of model-generated translations will be too short, but our goal in these trials is not to generate good translations, but to test out different model variations.

In all these trials we translate from Dutch to English to be able to use METEOR's synonym function.

We start by using models without an attention layer and experiment with such a layer only in our final preliminary trial.

4.1.1 Token Embeddings. We start by examining the impact of different token embeddings on model performance. We examine 3 different methods word embeddings:

[6] See https://huggingface.co/blog/how-to-generate for an excellent explanation of different text generation approaches.
[7] See: https://snowballstem.org/

(1) Word2Vec CBOW with hierarchical softmax,
(2) Word2Vec Skip-gram with hierarchical softmax and
(3) Embedding vectors learned by the model during regular training.

The Word2Vec models are trained using Gensim[13], and are trained in an unsupervised manner on the entire Dutch-English dataset, not only the 8% model training data.

We use a smaller model variant with a hidden-state size of 32 (for encoder/decoder LSTMs alike) for this first trial.

We observe the following results:

	corpus-level BLEU	mean METEOR
Word2Vec CBOW	1.198	11.222
Word2Vec Skip-gram	1.162	11.218
learned embeddings	1.136	11.655

We conclude that the choice of embedding-method does not greatly influence model performance. While it may make a small difference, there seem to be much more influential parameters, such as the parameters of the beam-search, or perhaps increasing model size.

We decide to use Word2Vec-CBOW vectors as token embeddings going forward, as they reduce the amount of parameters that need to be trained on each trial when compared to embeddings learned on the fly. This mean we can reuse Word2Vec vectors that have been trained once, and need to update less weights during the training of the actual model, slightly increasing training speed.

4.1.2 Character-Based Model. Now we scale up our model to use a hidden-state size of 128 (We refer to this model as the *base* model), which should in better translations. As mentioned before, we use the trained Word2Vec-CBOW vectors for token embeddings.

We compare this to a character-level model with the same hidden size of 128. For this model we use byte-based tokenizer: It simply assigns each byte of an UTF-8 encoded string a separate token, resulting in a vocabulary size of 256. ASCII-characters, which make up most of our data, are representable as a single byte/token. For <s>/</s> we designate the special Unicode control sequences *Start of Text* (U+0002) and *End of Text* (U+0003), which do not naturally occur in the Europarl dataset, and are mostly part of the Unicode standard for historical reasons; They are hardly ever used in written language.

As the vocabulary is much more restricted, the byte-based model uses a reduced embedding size of 32, transforming the 256-dimensional one-hot vectors into dense 32-dimensional representations.

Finally, we also need to adjust the parameters of the beam-search for a fair comparison: An English word is 5 characters long on average. Thus we use a maximum sequence length of 10 for the *base* model and a maximum length of 50 for the byte-based model.

These are our results:

	corpus-level BLEU	mean METEOR
token-based	0.833	11.542
byte-based	0.164	6.738

Clearly the token-based model performed much better. Let us examine an example to see why the byte-based model performed so poorly:

Actual translation	Model-generated translation
E467 is problematic in terms of assessing its harmlessness, because it contains extremely dangerous impurities.	The Commission of the political problems of this
This demand makes conciliation almost impossible and even affects the dignity of the parliamentary institution.	The Commission of the political problems of this

As can be seen already from this small random sample and by examining further examples, the model seems underfitted and apparently just learned to produce the same sequence on every input. We believe that the model may perform better when trained on more data, but it is clear that the token-based approach offers better quality with the constrained training data regardless:

Actual translation	Model-generated translation
What role, then, can and should central banks play in order to contribute to financial stability, to avoid financial crises and to provide assistance?	How can the EU 's internal market,
I shall now come back to three points.	I would also like to say that this is

While the translations are wrong, at least outputs are unique for every input and may capture the general "theme" of the text.

4.1.3 Attention Mechanism. In this trial we enable our attention head. Due to time constraints, we only train on 0.8% of the data. We would like to highlight again that in these trials it is important that we can observe if changes to the model produce *relative* downstream performance improvements.

These are our results:

	corpus-level BLEU	mean METEOR
no attention layer	1.122	11.222
with attention layer	3.615	20.353

We observe a large relative improvement of triple the BLEU-score when using an attention layer. The results of the METEOR metric also almost doubled. From manual inspection of random samples from the translations we observe that the model with attention actually manages to construct rough translations of short segments of the input. Meanwhile the model without attention also constructs grammatically adequate sentences, but these fail to be related to the given input sentence in any way.

4.2 Final Results

With the insights gained from our preliminary trials we train our final models, for which we enable the attention mechanism. We call this our *base-attn* model variant. To recap:

- It uses a unigram SentencePiece tokenizer with a vocabulary-size of 16000 trained on the respective language.
- It uses an embedding size of 256, which is trained beforehand as a Word2Vec-CBOW model. These embedding layers are not further updated during training.
- Both the encoder and decoder use one uni-directional LSTM-layer.
- It uses the same hidden state size of 128 as the *base* variant.
- We use Bahdanau-style additive attention for the decoder.

- The model has 10 715 904 parameters, of which 2 523 904 are trainable and 8 192 000 correspond to the two Word2Vec embedding layers.

We train the model on the whole 8% training split. When comparing the final results to those from the previous trial experiment we can observe that this clearly helped improve output quality.

Using our automatic evaluation metrics we observe the following results:

	corpus-level BLEU	mean METEOR
English to Dutch	11.527	33.024
Dutch to English	11.373	34.244

According to the automatic metrics our models performed very similarly on both languages.[8] We also examine a randomly picked sample of translations (and primarily show the results of the Dutch-to-English model as English is the language of this written report):

Actual translation	Model-generated translation
That doubt has now fallen over the position of the United Kingdom and its entitlement to Structural Funds because the United Kingdom government has not fully implemented or complied with the provisions of the habitats directive.	This is the case of the United Kingdom because it does not be taken into consideration, because it does not have been implemented or not in my country.
Mr President, the 60th anniversary of the end of the Second World War, which we recently commemorated, reminded us of the historical significance of the assistance we received from the United States.	Mr President, the latest 30th anniversary of the Soviet War has said on the historical significance of the United States.
What might perhaps be considered - and I myself do not know whether this already happens - would be for infant formula to be required to carry a warning, stating that it must be handled carefully, must not be left standing around open, and so on.	What could be said, if it may be said, if it is not the case of the quality of mothers, I know if it is not to be
Mijnheer de Voorzitter, ik zou mijn diepe misnoegdheid van de parlementaire delegatie en van de ten principale bevoegde parlementaire commissie, de REX-commissie, willen uitdrukken.	Mijnheer de fungerend voorzitter, ik wil de parlementaire delegatie bedanken voor de parlementaire delegatie en de Commissie.

As can be seen from these examples our models might work well enough, especially at the beginning of sentences, generally getting the content of the first 5 or so words correct. Afterwards it may keep the correct "general theme" of the text, but often introduces factual inconsistencies or starts to ramble and repeat a sequence of tokens over and over again. (We suspect that a higher number of beams in the search may produce more coherent sentences at this point.)

[8] It should be noted that the METEOR score for the Dutch to English translations is able to take synonyms into account, which may lead to a higher score when compared to the English to Dutch translation score, which is unaware of synonyms.

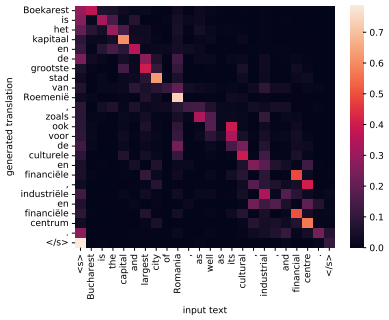

Fig. 8. an example where our model performs better

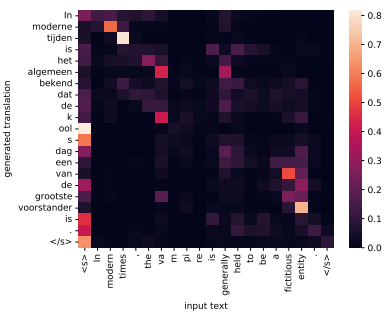

Fig. 9. an example where our model performes worse

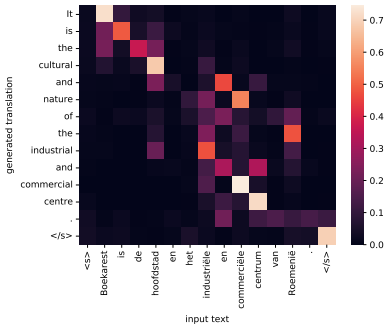

Fig. 10. attention weights for translation from Dutch to the pivot language (English)

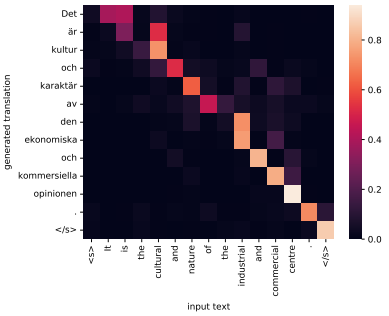

Fig. 11. attention weights for translation from the pivot language (English) to Swedish

As such we would expect our models to do much better in automatic scoring if we would only compare the first few words.

Last but not least we visualize the attention weights for some examples. We chose these examples from introductions of Wikipedia-pages, so they represent novel data the model has not seen yet. (However we did cherry-pick the most interesting translations out of examples.) Figure 8 and **??** show the results.

4.2.1 Dutch-Swedish Translation using English as a Pivot Language. We also developed a translation pipeline for translation from Dutch to Swedish using English as a pivot language. As we lack Dutch-Swedish test data we cannot evaluate the performance of the model using automatic means. Instead we decide to visualize the attention weights again, including the translation to the pivot-language (Figure 10) and back to the target (Figure 11).

5 CONCLUSION

Machine translation remains a challenging task. Models generating good and correct translations require large amount of time-consuming training, and evaluating a model on longer sequences necessitates inefficient beam-searches. Our LSTM-based models perform best when only considering the first few words of a translation but do not scale well to longer sentences, despite us using the attention-mechanism.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv: 1409.0473 [cs.CL].

[2] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, Ann Arbor, Michigan, (June 2005), 65–72. https://aclanthology.org/W05-0909.

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media Inc.

[4] François Chollet et al. 2015. Keras. (2015). https://keras.io.

[5] [SW] TensorFlow Developers, TensorFlow version v2.8.2, May 2022. DOI: 10.5281/zenodo.6574269.

[6] Alex Graves. 2012. Sequence transduction with recurrent neural networks, (Nov. 2012). arXiv: 1211.3711 [cs.NE].

[7] Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32, 4, (Dec. 2006), 485–525. DOI: 10.1162/coli.2006.32.4.485.

[8] Philipp Koehn. 2005. Europarl: a parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*. Phuket, Thailand, (Sept. 2005), 79–86. https://aclanthology.org/2005.mtsummit-papers.11.

[9] Taku Kudo and John Richardson. 2018. SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, (Nov. 2018), 66–71. DOI: 10.18653/v1/D18-2012.

[10] Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. COLING, Geneva, Switzerland, (Aug. 2004), 501–507. https://aclanthology.org/C04-1072.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, (July 2002), 311–318. DOI: 10.3115/1073083.1073135.

[12] Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14, 3, (Mar. 1980), 130–137. DOI: 10.1108/eb046814.

[13] Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, (May 2010), 45–50. http://is.muni.cz/publication/884893/en.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. Vol. 27, 3104–3112. https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates Inc. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[16] Matthew Watson, Chen Qian, Scott Zhu, François Chollet, et al. 2022. Kerasnlp. (2022). https://github.com/keras-team/keras-nlp.