Training and integration of neural machine translation with MTUOC

Antoni Oliver, Sergi Álvarez
Universitat Oberta de Catalunya (UOC)
`{aoliverg,salvarezvid}@uoc.edu`

Abstract

In this paper the goals and main objectives of the project MTUOC are presented. This project aims to ease the process of training and integrating neural machine translation (NMT) systems into professional translation environments. The MTUOC project distributes a series of auxiliary tools that allow to perform parallel corpus compilation and preprocessing, as well as the training of NMT systems. The project also distributes a server that implements most of the communication protocols used in computer assisted translation tools.

1 Introduction

1.1 The MTUOC project

MTUOC is a project that is being developed in the Arts and Humanities Department of the Universitat Oberta de Catalunya (UOC). The main objective of the project is to facilitate the training, use and integration of neural machine translation systems. It also provides tools for training statistical systems. Most of the software needed to train such systems is distributed in the form of complete toolkits under permissive free licenses. This makes, in principle, all this technology freely available to any professional, company or institution. However, the use of these *toolkits* brings with it a number of problems:

- *Technical skills*: A relatively high level of computer literacy is required to use these programmes. Knowledge of programming languages (e.g. Python) or scripting (e.g. Bash) is required. On the other hand, the documentation of these tools is not always sufficiently detailed and not always properly updated. This means that a lot of time is wasted in trial and error processes until the tools can be used effectively.

- *Integration*: The resulting translation engines cannot be easily integrated into existing professional translation workflows. Most tools provide access through some kind of API, usually in a server-client configuration. Some CAT tools provide access to some translation systems, but not all CAT tool - translation system combinations are available.

- *Hardware*: some high hardware requirements are necessary, especially for training systems. To train statistical systems, a lot of RAM is necessary. To train neural systems, one or more powerful GPUs (Graphical Processing Units) are essential.

The MTUOC project offers solutions for the first two problems. Regarding the *technical skills* it provides a number of easy-to-use Python and Bash scripts for corpus pre-processing and system training. All these scripts are well documented and can be easily adapted to specific needs. Regarding *integration* issues, a fully configurable server is provided that supports many communication protocols. This ensures compatibility with numerous CAT tools. For example, the server can use a neural engine based on Marian, but behave as a Moses server, so it can be directly integrated with OmegaT[1], a very popular free CAT tool. The project also provides a translation client that can

[1] `https://omegat.org/`

translate text and files in a variety of formats, including XLIFF. The client can also generate translation memories in TMX with the content of the machine translation, which facilitates integration into almost all CAT tools. Finally, with reference to the *hardware* problem, it should be noted that the computing requirements for training are much higher than for translation with already trained systems. Once a system is trained, it can be used for translation on any average consumer computer. In this way, many potential users can benefit from the various free engines distributed in the MTUOC project. To train customised systems, powerful GPU units are required. It is possible to purchase these units at affordable prices and install them in consumer computers. In addition, the UOC offers the possibility to establish technology transfer agreements for the training of customised systems at very competitive prices. This service is free of charge for NGOs. Since all the components of the MTUOC project are distributed under free license, any institution or company can offer customised training services using our components.

1.2 MT toolkits

As already mentioned, the MTUOC project provides auxiliary tools to work with the main neural (and statistical) machine translation *toolkits*. Although in principle the components can be adapted to work with any *toolkit*, the following have been considered and verified:

- Moses[2] (Koehn et al., 2007): is a *toolkit* for statistical MT systems training. Some of its pre-processing tools (such as tokenizers and truecasers) are still used for training neural engines.
- Marian[3] (Junczys-Dowmunt et al., 2018): is a toolkit for neural MT systems training. It is developed in C++ so it is very fast and efficient.
- OpenNMT[4] (Klein et al., 2017): it is also a *toolkit* for neural MT systems training. Two implementations are available: one based on Python and PyTorch (OpenNMT-py) and one based on TensorFlow (OpenNMT-tf). It includes a number of sub-projects as CTranslate2[5], a fast inference engine for models trained with OpenNMT, and Tokenizer, a tokenisation library with APIs for C++ and Python.
- ModernMT[6] (Bertoldi et al., 2018): until version 2.5 it provided the possibility of training both statistical and neural engines, but later versions only allow training of neural engines. It stands out for its ease of training and for providing adaptive machine translation. In this way, the translation provided will depend on the context in which the sentence to be translated is found.

1.3 Similar projects

There are a number of projects similar to MTUOC that also distribute neural engines and auxiliary programs that can be run on personal computers. One notable project is OpusMT[7]. (Tiedemann and Thottingal, 2020), which provides around 1,000 ready-to-run neural machine translation models to run on a machine translation server. Specifically, this project provides:

- Translation engines based on Marian-NMT.
- The engines are trained with corpora from the Opus-MT-Train collection[8]
- The provided scripts can be used to train your own systems.
- SentencePiece (Kudo and Richardson, 2018) is used in most engines for the calculation of subword units.
- Most engines have been trained using *guided alignment* based on word-level alignment using eflomal[9] (Östling, 2016).
- The related project Opus-Translator[10] provides a server that can run the trained engines.
- The related project OPUS-CAT[11] implements a plug-in for Trados Studio, a computer-

[2]http://www.statmt.org/moses/
[3]https://marian-nmt.github.io/
[4]https://opennmt.net/

[5]https://github.com/OpenNMT/CTranslate2
[6]https://github.com/modernmt/modernmt
[7]https://github.com/Helsinki-NLP/Opus-MT
[8]https://github.com/Helsinki-NLP/Opus-MT-train
[9]https://github.com/robertostling/eflomal
[10]https://github.com/Helsinki-NLP/OPUS-translator
[11]https://github.com/Helsinki-NLP/OPUS-CAT

assisted translation tool widely used in professional environments.

The models trained in the Opus-MT project are also distributed in HuggingFace[12] (Wolf et al., 2019), further facilitating the use of these engines.

Softcatalà[13] is a non-profit association whose main objective is to promote the use of the Catalan language in computing, Internet and new technologies. This association leads numerous projects for the localisation of free software projects into Catalan. To facilitate this localisation, they use machine translation engines and for some years now they have been training their own neural engines, which they distribute under a free licence (Irigoyen et al., 2020). At the time of writing, the following engines including Catalan were offered with the following languages in the two directions: German, Spanish, French, Dutch, Italian, Portuguese and Spanish. The models are trained with OpenNMT and can be used with the components of this toolkit and also with CTranslate2. In addition Softcatalà provides a set of tools to use the trained models:

- A tool for processing texts in various formats into the OpenNMT input text format.
- A server program that provides an API to translate through a web service.
- Tools to directly translate text files or files in PO format.

2 Components of the MTUOC project

The MTUOC project offers a series of programs and scripts covering all steps from the creation of parallel corpora, their cleaning and corpus combination, to a program that allows to evaluate the trained systems using the most common automatic metrics. It also offers a series of scripts for corpus preparation and pre-processing and training using the toolkits presented in section 1.2. It also provides a complete server that is able to communicate with the servers provided by the main toolkits and to communicate with the client using a large number of protocols. The programs and scripts of the MTUOC project are programmed in Python version 3 or are Bash scripts. All components can be downloaded from the project's Github page and are distributed under a free licence. In this section we present a brief description of each of these components.

2.1 Programs for the creation of parallel corpora

Training neural machine translation systems requires large and good quality parallel corpora. While there are methods and toolkits for unsupervised training from monolingual corpora, supervised systems using parallel corpora achieve much higher levels of quality. Systems can be trained with parallel corpora available in a number of repositories, including OPUS[14] (Tiedemann, 2012). However, in professional environments it is common to have large amounts of parallel texts for specific topics and clients, usually in some translation memory format (TMX, SDLTM, etc.) or in the form of completed translation projects (often in the standard XLIFF format). To take advantage of these resources in system training, the MTUOC project provides a number of utilities:

- `TMXdetectlanguages`: detects the language codes present in a given TMX file or in all TMX files in a given directory.
- `TMX2tabtxt`: converts a TMX file or all TMX files in a given directory to tabular text. It supports more than one possible language code for the source language and the target language.
- `sdltm2tabtxt`: converts one SDL-Trados (SDLTM) memory or all memories in a given directory to tabular text format.
- `XLIFF2tabtxt`: converts a translation project interchange file (XLIFF) or all files in a given directory to tabular text format.

Another very common situation in professional environments is to have a large set of original and translated documents of a given subject and client, although the corresponding translation memories are not available. The MTUOC project provides a series of auxiliary tools that facilitate the alignment of these documents with hunalign[15] (Varga et al., 2007):

- a program for segmenting using SRX files (*Segmentation Rules eXchange*). It allows to

[12]https://huggingface.co/
[13]https://www.softcatala.org/
[14]https://opus.nlpl.eu/
[15]https://github.com/danielvarga/hunalign

segment a single file or all files in a directory. The program can include the paragraph marks (<p>) required by hunalign.

- a series of bilingual dictionaries for various language pairs in the format required by hunalign. These dictionaries have been created from the transfer dictionaries of the Apertium machine translation system Apertium[16] (Forcada et al., 2011).

- A program for creating the alignment script with hunalign to speed up the alignment of large numbers of documents.

- A program to select aligned segments above a certain value of the confidence index provided by hunalign.

When the files to align are parallel, that is, there is a quite good correspondence between the source and target documents, using hunalign we can get very accurate alignments. It can handle missing segments in one of the languages, and it can also handle different relations between source and target segments (1 to many and many to one).

In some cases, we can get a set of documents in the source language and a set in the target language. These documents can be translation equivalents but we don't know which document is the translation, as the names doesn't' match. The documents can talk about similar concept but they are different, and some parallel segments might exists. In these cases, techniques for mining parallel segments in comparable corpora (Schwenk, 2018) may be very productive. The MTUOC project includes a program that can perform this mining using SBERT (Reimers and Gurevych, 2019). The program provides you with parallel segments sorted by a confidence score. A visual inspection of the results is usual enough to select the lower score limit and reject parallel segments with a lower score.

2.2 Parallel corpus cleaning program

When obtaining or creating a parallel corpus, it is useful to clean up the corpus. MTUOC provides a parallel corpus cleaning program (`MTUOC_clean_parallel_corpus.py`) that can perform the following actions:

[16]`https://www.apertium.org/`

- Replace the typographical apostrophe with the standard apostrophe (`norm_apostrophe`).

- Remove HMTL and XML tags (`remove tags`).

- Replace HTML/XML entities with their corresponding characters (`unescape_html`).

- Remove parallel segments with empty segments (`remove empty`).

- Remove segments that are too short by setting a minimum number of characters (`remove_short`).

- Remove segments where the segment in the source language and the segment in the target language are the same (`remove_equal`).

- Remove the segments in which the percentage of numbers in either language is higher than a certain value (`remove_NUMPC`).

- If you want to preprocess to train Moses systems, replace the characters [and | in the corpus by their corresponding HMTL entities (`escapeforMoses`).

- Remove segments containing the specified elements from a file (`stringFromFile`).

- Remove segments that match at some point the regular expressions indicated in a file (`regexFromFile`).

- Check that the language of the source segment is correct (`vSL`).

- Check that the language of the target segment is correct (`vSL`).

- Check that the language of the target segment is not a certain language (`vNOTL`).

- The number of languages detected by the automatic language detection algorithm can be limited by using the `vSetLanguages` option.

- Remove segments written in upper case (`noUPPER`).

2.3 Parallel corpora rescoring

Once we have cleaned the corpus, we have a set of segment pairs that do not have any of the selected problems. Anyway, both segments in the segment pair can be clean, but they may not be translation equivalents. It is possible to calculate a score that provides an idea of the translation equivalence between the source and target segments in the segment pair. This can be achieved calculating the sentence embedding of the source and target segment using a multilingual model, and calculating a distance measure, as the cosine distance, for example.

The toolkit provides a program that performs two actions:

- Language detection of the source and target segments, using fasttext. This tool offers two interesting features: it returns the detected language along with a detection confidence score and users can easily train their own language detection models.
- Scoring all the segments with the cosine distance of the sentence embedding representation calculated using a multilingual model, LaBSE (Feng et al., 2022) by default.

After the scoring process is completed, we use a companion program to select the parallel segments matching a series of conditions: languages detected and language detection confidence score; and a minimum confidence score based on the cosine distance.

2.4 Parallel corpus combination program

Another common situation in professional environments is to have a large number of translation memories and original and translated texts which, once processed with the programs described in section 2.1, generate a parallel corpus of insufficient size for system training. The MTUOC project provides a series of programs to select from a parallel corpus the segments most similar to those of another parallel corpus. For example, we have a parallel corpus A of the pair L1 - L2 but it is not large enough to train a system. We also have a much larger corpus B for the same language pair. We are interested in selecting from corpus B the segments that are most similar to those of corpus A. The program calculates the L1 language model from corpus A and from this language model it selects the most similar segments from corpus B (verifying the perplexity of the L1 segments of corpus B with respect to the language model). The program also divides the A+B corpus into three parts:

- training (train): will contain segments from A and B.
- validation (val): will contain only segments from A.
- evaluation (eval): will contain only segments from A.

In this way, corpus B extends only the training corpus, as the validation is carried out exclusively with segments of corpus A.

The same program can be used in the case we only have a monolingual corpus A and a parallel corpus B. We can select the most similar parallel segments from B using the corpus A. In this case, when dividing the resulting corpus into training, validation and evaluation, all these sets will contain only segments from corpus B.

2.5 Script for the preparation of parallel corpora

MTUOC distributes all the necessary components to prepare the parallel corpus for further preprocessing. We have made a distinction between these two phases, preparation and preprocessing, since the preparation steps are (or at least can be) common for the training of statistical and neural systems. In contrast, the preprocessing step will be different. The components for corpus preparation are:

- Tokenisers for the following languages: Aragonese (arg), Asturian (ast), Catalan (cat), German (deu), English (eng), French (fra), Galician (gal), Italian (ita), japanese (jap), Portuguese (por), Russian (rus), Sardinian (srd), Spanish (spa) and Chinese (zho). A generic tokeniser (gen) is also distributed which can be used for other languages. In addition, a pseudo-tokeniser is distributed for Chinese (zho-pseudo) which simply separates all characters in the file by whitespace and another Chinese tokeniser based on the jieba footnote library[17] (zho_jieba). These tokenisers have different modes of operation and can

[17] https://github.com/fxsjy/jieba

represent tokens with either *joiners* or *splitters* to facilitate subsequent detokenisation. Tokenisers also function as detokenisers. Tokenisers give the option to separate digits from numerical expressions, a common practice in neural engine training. New language-specific tokenisers for other languages will be added in the future.

- Truecasers, both the training program and the program that carries out the truecasing process. The truecaser can be trained with a corpus and a dictionary of language forms.
- Programs to replace emails and URLs with configurable codes.
- Program for splitting the corpus into fragments of different number of segments. It is useful to divide the corpus into training, validation and evaluation fragments.

All these steps can be performed with a single program that is fully configurable by means of a yaml configuration file, which can be easily edited in any text editor.

2.6 Scripts for the preprocessment of parallel corpora

In this step we perform a series of operations that depend on the type of engine we want to train. Three different programs are provided, which are configured by means of yaml files.

- For statistical engines, the steps of replacing numerical expressions with codes are performed and the characters [and | are replaced by their corresponding entities.
- For neural engines using BPE (Byte Pair Encoding) (Sennrich et al., 2016) for subword calculation using the subword-nmt algorithm[18].
- For neural engines using SentencePiece[19] (Kudo and Richardson, 2018) for subword computation.

2.7 Training scripts

Scripts for training translation systems are provided for several toolkits, namely

- Moses: scripts are provided to perform all the training in one step or to perform the different steps individually: training (of the language and translation model), SALM (Suffix Array tool kit for empirical Language Manipulations), optimisation and binarisation.
- Marian: scripts for s2s and transformer systems.
- OpenNMT: scripts for transformer type systems.

2.8 MTUOC server

The models we train, whether statistical with Moses or neural with Marian or OpenNMT, can be implemented in a program that works as a translation server, that is, as a program that waits to receive segments to be translated and returns these translated segments. Figure 1 shows the operation scheme of the client-server configuration. The client can be the MTUOC-Translator (see section 2.9) or a CAT tool. This client program sends the segments to the MTUOC server. The segments can be sent and received in different protocols, which allows this machine translation system to be compatible with various computer-assisted translation tools. Specifically, it can use the following protocols:

- MTUOC: a simple protocol specific to the project. We have developed a plug-in for Trados Studio 2019, 2021 and 2022.[20]
- Moses: the same protocol used by the server provided by this toolkit. It can be used, for example, with OmegaT and Trados Studio 2017 and 2019.
- ModernMT: the same protocol as the server provided by this toolkit. It can be used, for example, with Okapi tools[21] (Tikal and Rainbow).
- OpenNMT: the same protocol as the server provide by this toolkit.
- NMTWizard: the same protocol used by this server.[22].

[18] `https://github.com/rsennrich/subword-nmt`
[19] `https://github.com/google/sentencepiece`
[20] `https://github.com/aoliverg/MTUOC-Trados-plugin`
[21] `https://okapiframework.org/`
[22] `https://github.com/OpenNMT/nmt-wizard`

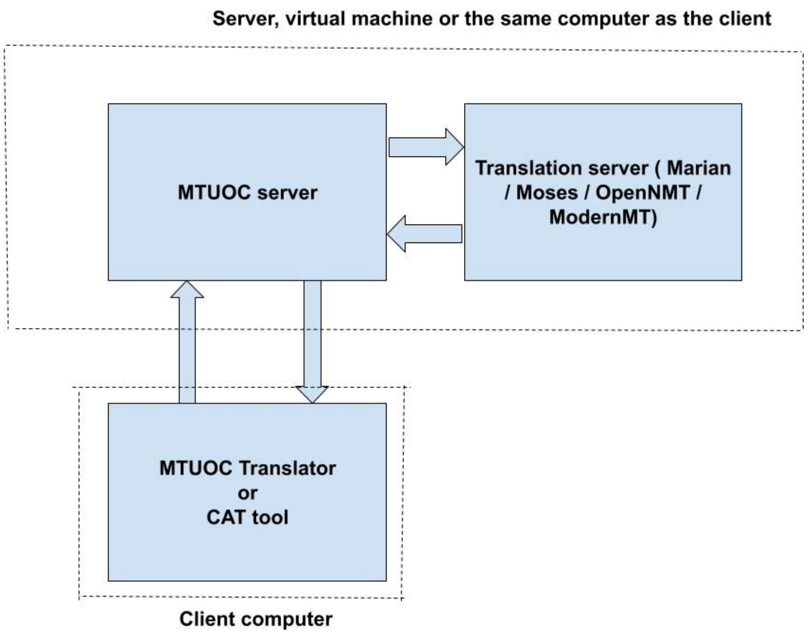
Server, virtual machine or the same computer as the client
MTUOC server
Translation server (Marian / Moses / OpenNMT / ModernMT)
MTUOC Translator or CAT tool
Client computer

Figure 1: Client-Server configuration in MTUOC engines

The client sends the segment as it is, and the MTUOC server preprocesses it (e.g. tokenises, truecases and SentencePiece) and sends it to the appropriate translation server (e.g. marian-server). When the MTUOC server receives the translated segment it post-processes it (e.g. detokenises, detruecases and undoes the SentencePiece subword) and sends it back to the client program.

Do not confuse the protocol that the MTUOC server will use with the translation system it will use. For example, we may have a neural engine trained with Marian but we want to use it with a tool that is only compatible with the Moses server protocol. In this case we can run the MTUOC server in Moses mode and it will be compatible with the desired tool, but it will actually be translating with a Marian neural engine.

MTUOC-server can be run in personal computers or in physical or virtual servers. Running the server in your own computer or server ensures you full confidentiality, as no information goes out from your premises. No GPU is required to run the server, but using GPUs will speed up scientifically the translation speed. Having your own engines integrated in MTUOC-server allows you to run as much instances as necessary in case of very heavy use. For very heavy use scenarios, a wsgi server such as gunicorn[23] can be used along with MTUOC-server.

[23]`https://gunicorn.org/`

2.9 MTUOC Translator

The MTUOC project provides a client program with a simple user interface that allows to translate segments and files in different formats. It also provides a version without graphical interface that is useful in the evaluation process of trained engines.

2.10 Tag processing in MTUOC

The MTUOC server includes an automatic XML/HTML tag retrieval algorithm. The training of the engines is carried out with corpora in which all tags have been removed and the translation is carried out with untagged segments. If the segment to be translated contains tags, these are removed. Once the segment has been translated, the tags are retrieved from the original segment with tags, from the untagged translation and from the alignment information provided by the translation engine. Statistical engines trained with Moses are able to return reliable alignment information. Neural engines, especially if they are of the transformer type, do not return reliable alignment if they are not trained with *guided alignment*. If trained with *guided alignment* they are able to return quite accurate alignment information, but this information will be relative to the sub-words.

The tag retrieval algorithm is able to retrieve tags even if subword-based information is available.

2.11 Evaluation program

The project distributes an evaluation program that provides several automatic evaluation metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), WER (Word Error Rate) (Nießen et al., 2000), %Ed. dist., TER (Translation Error Rate) (Klakow and Peters, 2002) and COMET[24] (Reimers and Gurevych, 2019).

To facilitate the use of this program, it is provided in two versions: one with a graphical user interface and one to be used from the terminal. The program calculates global values and it can also calculate detailed values by segment.

3 Free MT engines

The MTUOC project provides a growing number of freely downloadable MT engines. Most of the engines are based on Marian with a transformer configuration and guided-alignment. At the time of writing this paper, the following engines were available:

- General language: spa↔cat, eng↔spa, eng↔cat, fra↔spa, fra↔cat, rus↔spa, rus ↔cat
- International relations (using UNPC corpus): ara↔spa, eng ↔spa, fra ↔spa, rus ↔spa, zho ↔spa. Catalan version with synthetic corpora through Spanish using Apertium: ara↔cat, eng ↔cat, fra ↔cat, rus ↔cat, zho ↔cat.
- Patents (using EuroPat corpus): eng ↔spa, eng ↔fra, eng ↔deu.
- Legislation (using the DOGC corpus): cat ↔spa

  The complete and up-to-date list of MT engines can be seen in the MTUOC project wiki.[25]

4 Conclusions and future work

In this paper we have presented the MTUOC project, aiming to make the process of training and integrating NMT systems easier. With the components of this project, any professional or translation company will be able to use their own MT systems. One of the goals of the project is to make the developed tools usable for most users, regardless of their technical skills. We are now working on converting the scripts into programs with GUI interfaces to make them even easier to use.

This set of tools is being actively used in teaching activities at the bachelor and master levels at the UOC, and students with no previous knowledge of the use of terminal and scripts are able to perform all the processes involved in the training of statistical and neural systems. The tools are also being used in production environments, where custom NMT systems have been trained and used in translation projects.

In the future we plan to train and make freely available more NMT systems. We also want to offer NMT systems for low resourced language pairs, starting for some Romance languages in Spain: Asturian, Aragonses and Aranese.

Acknowledgments

This work is partially supported by the project *TAN-IBE: Neural Machine Translation for the romance languages of the Iberian Peninsula*, founded by the Spanish Ministry of Science and Innovation Proyectos de generación de conocimiento 2021. Reference: PID2021-124663OB-I00.

References

Bertoldi, Nicola, Davide Caroselli, and Marcello Federico. 2018. The ModernMT project. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation.*

Doddington, George. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Cooccurrence Statistics. *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145.

Feng, Fangxiaoyu, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891.

[24]https://unbabel.github.io/COMET/html/index.html

[25]https://github.com/aoliverg/MTUOC-project/wiki

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

Irigoyen, Marc Riera, Xavier Ivars Ribes, Pere Orga Esteve, Joan Montané Camacho, Jordi Mas Hernández, and Artur Vicedo Cremades. 2020. Softcatalà: nous reptes per garantir la vitalitat del català a les tecnologies. *Revista de Llengua i Dret*, (73):146–153.

Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July. Association for Computational Linguistics.

Klakow, Dietrich and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, page 67–72.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Kudo, Taku 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*, pages 66–71.

Nießen, Sonja, Franz Josef Och, Gregor Leusch, Hermann Ney, et al. 2000. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of the LREC Conference Athens, Greece, 2000*. Citeseer.

Östling, Robert. 2016. Efficient word alignment with markov chain monte carlo. *The Prague Bulletin of Mathematical Linguistics*, (106):125–146.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wj Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Reimers, Nils and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.

Schwenk, Holger. 2018. Filtering and mining parallel data in a joint multilingual space. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–234.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. 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.

Tiedemann, Jörg and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

Tiedemann, Jörg. 2012. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218.

Varga, Dániel, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. In *Recent Advances in Natural Language Processing IV*, pages 247–258. John Benjamins.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.