Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

AN ONLINE REPOSITORY OF PYTHON RESOURCES FOR TEACHING MACHINE TRANSLATION TO TRANSLATION STUDENTS

Ralph Krüger

TH Köln (University of Applied Sciences)

Abstract

This paper presents an online repository of Python resources aimed at teaching the technical dimension of machine translation to students of translation studies programmes. The Python resources provided in this repository are Jupyter notebooks. These are web-based computational environments in which students can run commented blocks of code in order to perform MT-related tasks such as exploring word embeddings, preparing MT training data, training open-source machine translation systems or calculating automatic MT quality metrics such as BLEU, METEOR, BERTScore or COMET. The notebooks are prepared in such a way that students can interact with them even if they have had little to no prior exposure to the Python programming language. The notebooks are provided as open-source resources under the MIT License and can be used and modified by translator

training institutions which intend to make their students familiar with the more technical aspects of modern machine translation technology. Institutions who would like to contribute their own Python-based teaching resources to the repository are welcome to do so.

Keywords: translation technology, machine translation, natural language processing, translation didactics, Jupyter notebooks, Python programming

1. INTRODUCTION

Fuelled by advances in artificial intelligence (AI) and AI-driven natural language processing (NLP) research coupled with the availability of large volumes of digital training data such as translation memories (TM) or parallel corpora, neural machine translation (NMT) has become the state-of-the art machine translation (MT) architecture in recent years. Due to the considerable increase in output quality of NMT compared to previous MT architectures (for an overview of corresponding studies, see, e.g., Jia et al., 2019, p. 60), NMT systems are increasingly employed in the professional translation process, where translators can draw on the output of these systems alongside traditional resources such as TMs or termbases. It stands to reason that translator training institutions have to react to this growing prominence of neural

machine translation in the translation profession by giving MT teaching a higher priority in their curricula. The call for teaching extended MT competences comes, for example, from the translation competence framework of the European Master's in Translation (EMT) programme. In its revised version from 2017, this framework requires graduates of MA programmes in translation to "[m]aster the basics of MT and its impact on the translation process" and to be able to "[a]ssess the relevance of MT systems in a translation workflow and implement the appropriate MT system where relevant" (EMT, 2017, p. 9). In a similar vein, O'Brien & Ehrensberger-Dow (2020, p. 146) make a general call for developing an adequate level of "MT literacy" in the wider translation community, which means "knowing how MT works, how it can be useful in a particular context, and what the implications are of using MT for specific communicative needs".

Naturally, university translation programmes aiming to develop such MT literacy in their students will focus mainly on the linguistic dimension of MT. They will teach students the general advantages and shortcomings of MT, its (un)suitability for translating different text types and genres, how to perform manual MT quality estimation, how to pre-edit texts for MT or how to

post-edit the output of MT systems according to different quality requirements. Often, students will also be provided access to a graphical user interface(GUI)-based MT system such as *KantanMT*, where they can use their own training data or precompiled training data to train an MT engine, evaluate its quality using manual evaluation or a range of automatically calculated quality scores and then integrate this engine into translation environment tools such as *Trados Studio*. While systems such as KantanMT arguably allow students to develop competences in some more technical aspects of MT, they still shield them from the deeper technical issues involved in handling MT systems. It could certainly be argued that this is in fact a desirable state of affairs since mastering the linguistic side of MT as well as gaining a modest understanding of its technical dimension represents an adequate level of MT literacy for translation students. Also, these competences are basically what is called for in the EMT competence framework discussed briefly above. On the other hand, the argument could be made that – in light of the growing pervasiveness of machine translation in the translation profession – translation students should master both the linguistic and the technical dimension of MT to a high degree in order to be able to act as "text and translation technology experts" (Christensen et al., 2017, p. 19) for their

future employers or clients. To date, there seems to be no consensus in translation didactics on how much of the technical complexity of MT translation students should be exposed to (however, see the insightful reflections in Kenny & Doherty, 2014, pp. 288–289), and different translator training institutions will certainly arrive at different answers to this question depending on their tradition, the focus of their translation programmes, the university context they are embedded in, the areas of specialization of their lecturers, etc.

This paper presents an online repository of Python-based teaching resources, which can be used by translation teachers who would indeed like to teach their students the technical dimension of machine translation in a more detailed and exhaustive manner. These teaching resources require little to no programming skills on the part of the students, making them ideal didactic instruments for a gentle introduction to the technical dimension of MT in a translation-oriented learning context.

2. THE TECHNICAL DIMENSION OF MACHINE TRANSLATION

As mentioned in the introductory section, systems such as KantanMT expose students to a moderate degree of MT complexity while shielding them

from its deeper technical aspects. For example, students need just a few mouse clicks in order to create a new MT engine, drag and drop the training data and start the training process. However, data preparation for MT training actually requires a more complex pipeline which students will remain ignorant of when training an MT engine with systems such as KantanMT (again, some would argue that this is indeed desirable). An actual data preparation pipeline for MT training involves tasks such as filtering the data[1], creating a subword vocabulary of the training data, converting the data into such subword units and splitting the data into training, development and test sets (see, e.g., Buj et al., 2020, p. 331). Also, before being fed into an MT system for training or translation, natural language data have to be converted into numeric vector representations (so-called *word embeddings*, see Mikolov et al., 2013) which the system can actually process. Furthermore, when setting up an MT system, different basic architectures such as recurrent neural networks (RNNs) or transformers are available, each with various configurable hyperparameters (e.g., network layers, vector

[1] For example, deleting empty cells, removing HTML or other markup tags, tokenizing the data, removing duplicates or segments with identical source and target sides or removing segments where there is too much of a length difference between source and target sides.

dimensionality, learning rate, training batch size, etc, see OpenNMT, 2017). Returning to the GUI-based system KantanMT, after engine training is completed, the system provides a range of traditional automatic MT quality scores[2] that students can use to evaluate the quality of their MT engines. However, students will get only a vague idea of what these scores actually mean, how they are computed and what their relative advantages and shortcomings are. Also, with the rise of neural networks in NLP, a new generation of MT quality scores has emerged which does not rely on the string-matching operations employed by traditional scores but rather on the similarity of word embeddings in high-dimensional vector space. Examples of such embedding-based scores would be *BERTScore* (Zhang et al., 2020) or *COMET* (Rei et al., 2020). In light of the high relevance of the overall neural paradigm for NLP and NLP-assisted professional translation, it would certainly benefit students to gain an understanding of these modern MT quality scores as well, regardless of whether they are already implemented in GUI-based MT systems such as KantanMT or not.

The Python-based MT training resources presented in the next section address several of these

[2] For example, *BLEU* (Papineni et al., 2002) or *TER* (Snover et al., 2006).

technical aspects of machine translation sketched above. They give both lecturers and students more flexibility in exploring various aspects of MT such as automatic MT quality scores, making them independent from specific systems such as KantanMT, Trados Studio, etc., which may or may not have the desired features implemented in a way that students or lecturers require.

3. JUPYTER NOTEBOOKS FOR MACHINE TRANSLATION TEACHING

The MT training resources were set-up in the form of Jupyter notebooks, which are provided in the following GitHub repository: https://github.com/ITMK/MT_Teaching.

The didactic value of Jupyter notebooks in higher-education teaching has been widely recognized (see, e.g., Barba et al., 2019) and they are used in diverse fields such as mathematics, computer science, computational linguistics, data science or the digital humanities. However, to the author's knowledge, Jupyter notebooks are not commonly used yet in teaching translation technology or NLP to translation students.

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

3.1. Basic Information on Jupyter Notebooks

Jupyter notebooks are interactive, web-based computing resources for "developing, documenting, and executing code, as well as communicating the results" (Jupyter Team, 2015). These notebooks combine two components, a web application for authoring notebook documents and the actual notebook documents, which can contain explanatory documentation, executable code cells and other resources (Jupyter Team, 2015). As Barba et al. (2019, p. 10) point out from a didactic perspective, the combination of documentation and code provided in Jupyter notebooks offers "[t]he opportunity of intermingling computation into a narrative, creating a conversation with data [that] is a powerful and effective form of communication". The documentation in Jupyter notebooks is written in markdown format and usually contains explanatory information on a certain topic. The author can format this documentation in various ways, integrate hyperlinks, display mathematical notation, etc. The code blocks are written in the programming language chosen for the notebook.[3] They usually contain commented lines of code which the notebook user can execute.

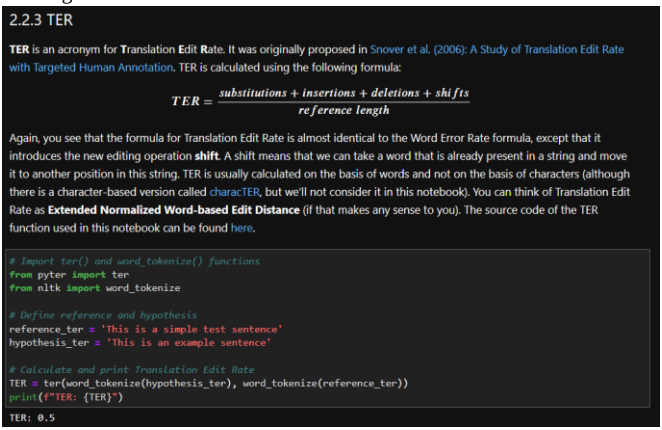
[3] The notebooks presented here are based on Python, but Jupyter notebooks support a wide range of other programming languages such as *R* or *Julia*.

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

3.2. Example of a Jupyter Notebook

Figure 1 shows an excerpt of a Jupyter notebook on calculating traditional MT quality scores based on string-matching operations. The section depicted in figure 1 is concerned with calculating the translation edit rate (TER):

Figure 1: Excerpt of a Jupyter notebook on MT quality score calculation, showing a markdown section and an executable code cell



The upper part of figure 1 shows a markdown section, which introduces the notebook user to the translation edit rate. This section contains some bold formatting, a hyperlink to the original TER paper, the TER formula in mathematical *MathJax* notation, a link to a paper introducing a character-based version of TER and another link to the

source code of the TER function used in this notebook. In this case, the source code is provided by the external *pyter* package (Tanaka & Vanroy, 2019), which can be installed and used as part of the notebook. The documentation can be written as explicitly or implicitly as the notebook author desires, but in a didactic context where these notebooks are used to introduce students to unfamiliar topics (in an unfamiliar environment at that), a more verbose documentation is probably desirable. Form and content of the documentation will also depend on how the notebook relates to other MT teaching resources provided to students. For example, the notebooks presented in this paper are intended to supplement an extensive *PowerPoint* presentation on the different dimensions of NMT in the course *Processes of Machine Translation* in the MA in Specialised Translation programme at the Institute of Translation and Multilingual Communication at TH Köln, Germany. This presentation introduces students to the basic concepts of automatic MT quality scores and discusses several of these scores along with their advantages and shortcomings. The notebook illustrated in figure 1 provides additional information on these scores, so the notebook documentation has to be seen in this particular teaching context. Lecturers aiming to employ these notebooks in their individual MT teaching contexts

may want to tailor the notebook documentation to these contexts.

The code cell in the lower part of figure 1 contains various commented blocks of code, which are executed when the notebook user runs this code cell. In this case, the code cell first imports the functions necessary to calculate the translation edit rate. The source code for TER calculation is included in the *ter()* function, which is part of the pyter package (visible from the command *from pyter import ter*). In addition to the *ter()* function, the code cell also uses the *word_tokenize()* function from the *Natural Language Toolkit* (NLTK, Bird et al., 2009) (*from nltk import word_tokenize*). This function is used to tokenize the input strings into a list of individual words, which is the input format required by the *ter()* function. The next code block defines the two sentences which will be used for TER calculation. In MT research, the hypothesis (stored in the variable *hypothesis_ter* in this case) is the output of a machine translation system and the reference (stored in the variable *reference_ter*) is a human reference translation. The strings 'This is a simple test sentence' and 'This is an example sentence' are just two random placeholder sentences which can be modified by students to calculate TER scores for their own hypothesis and reference sentences. The last code block calculates

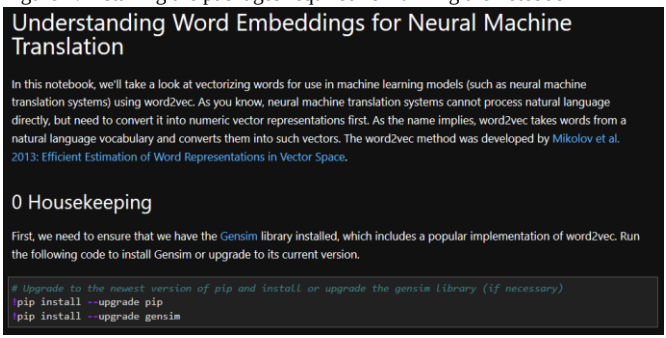
the translation edit rate for the strings defined previously, by tokenizing them using the commands *word_tokenize(hypothesis_ter)* and *word_tokenize(reference_ter)* and feeding them into the *ter()* function, which performs the actual TER calculation. The *print()* function in the last code line then prints the calculated score, and the string *TER: 0.5* below the code cell is the output produced when executing the code cell.

Jupyter notebooks such as the one presented here can be set up in such a way that interacting with them requires little to no programming skills on the part of the students (this is the case for all notebooks provided in the online repository), so that they can focus on the actual domain knowledge covered in the notebooks. However, if students do possess a certain level of programming skills – probably because basic Python coding is part of their translation curriculum or because they acquired these coding skills out of personal interest – they can hone these skills when interacting with the notebooks. For example, students could restructure code blocks or change the behaviour of certain code cells according to their requirements, they could add code cells on their own or they could analyse the source code of the various functions used in the notebook to see how the

actions performed by these functions are implemented in Python code.

At the beginning of each notebook, there is a 'housekeeping' step, which involves installing or upgrading various packages required for running these notebooks. Figure 2 below shows an example of such a housekeeping step in a notebook on exploring word embeddings for NMT:

Figure 2: Installing the packages required for running the notebook



Running the code cell in figure 2 will first ensure that the latest version of the Python package installer *pip* is installed. Then, this package installer is used to install or upgrade the *Gensim* package, which includes a popular implementation of *word2vec*, a technique used to convert words into high-dimensional word vectors. Once this housekeeping step is completed, students can

proceed to interact with the actual content of the notebook.

3.3. Computing Environments for Running Jupyter Notebooks

There are several ways to run the Jupyter notebooks provided in the GitHub repository, each with their own advantages and disadvantages. For example, the notebooks can be run on top of a local Python installation (preferably an *Anaconda Python* distribution since this comes preinstalled with most of the packages required for running the notebooks) on the students' own computers or on the computers of the university's computer lab. Also, the notebooks can be run on university-owned and maintained servers or in cloud environments such as *Kaggle* (Kaggle, n. d.) or *Colaboratory* (*Colab*, Google, n. d.). Barba et al. (2019, pp. 53–57) provide an excellent overview of the different options for providing students with access to Jupyter notebooks and discuss in detail the individual advantages and disadvantages of these options.

The MT notebooks provided in the GitHub repository presented in this paper can be downloaded and provided using any of these options. In addition to the GitHub repository, they

are also provided in a *Binder* environment, which is "an open source web service that lets users create sharable, interactive, reproducible environments in the cloud" (Project Jupyter et al., 2018, p. 113). Binder creates a Docker image of the GitHub repository and provides an executable environment in which all the packages necessary for running the Jupyter notebooks are preinstalled and students can start to interact with the notebooks right away. Since all necessary packages are already preinstalled, students can skip the housekeeping step described above, making Binder one of the most accessible options for running Jupyter notebooks. However, the Binder sessions are only temporary, meaning that all changes students make in a notebook during a Binder session will be lost when the session is terminated.

The GitHub repository also contains a link to the notebooks provided in a Colab environment, which is "a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs" (Google, n. d.). In Colaboratory, students can also run the notebooks directly and changes or additions made to the notebooks can be saved permanently and made available for subsequent sessions. However, Colaboratory does not provide an environment in which all packages required to run the notebooks

are already preinstalled, meaning that students will have to perform the housekeeping step before they can start to work with the notebooks. Also, new packages are not saved by default, meaning they may have to be reinstalled when opening another session. One of the major advantages of Colab is that users have free access to high-performing computing resources in the form of graphical processing units (GPUs). Such GPUs are necessary for performing resource-intensive machine learning tasks such as training word embedding models, neural language models or NMT systems. It is therefore recommendable to run notebooks performing such resource-intensive tasks in a Colaboratory rather than in a Binder environment.

3.4. Current Content of the Repository

At the time of writing this article (March 2021), the GitHub repository contains Jupyter notebooks for exploring word embeddings in the context of NMT, for computing traditional MT quality scores such as BLEU, METEOR (Bannerjee & Lavie, 2005), chrF (Popović, 2015), and TER and for calculating modern embedding-based MT quality scores such as BERTScore and COMET. Further notebooks are currently being prepared, for example, a notebook illustrating a typical MT training data preparation pipeline, a notebook for

training a neural machine translation system based on the *OpenNMT* toolkit (Klein et al., 2020) and a notebook for exploring neural language models such as BERT (Devlin et al., 2019) or domain-adapted BERT variants such as SciBERT (Beltagy et al., 2019). These neural language models, which are based on the transformer architecture originally developed in the context of neural machine translation (Vaswani et al., 2017), have achieved new state-of-the-art performance in NLP tasks such as text summarization and question answering and may have a role to play in the professional translation process of the future. Therefore, translation students should also be at least roughly familiar with these language models.

4. CONCLUDING REMARKS

This article presented an online repository of Python resources in the form of Jupyter notebooks, which are tailored to teaching the technical dimension of machine transition to students of translation programmes. Interacting with these notebooks requires no programming skills on the part of the students, so that they can focus solely on the technical MT concepts covered in these notebooks. In the case that students do have prior knowledge of Python programming, because it is part of their curriculum or they acquired it out of

personal interest, they can hone these skills when interacting with the notebooks. The notebooks can be downloaded and provided in various ways or they can be run directly online in a Binder or a Colaboratory environment. The notebooks are provided as open-source didactic instruments under the *MIT License* (Open Source Initiative, n. d.). The repository will be extended with further notebooks in the future, and lecturers interested in exposing their students to more technical MT concepts can use these notebooks or contribute their own notebooks to the repository.

As stated in the introductory section, the growing pervasiveness of neural machine translation in the professional translation process requires students of translation programmes to develop a reasonably high degree of MT literacy if they aspire to a successful career in the digitised translation industry. Naturally, they are first and foremost predisposed to become experts in the linguistic aspects of MT, but they will certainly also profit from being exposed rigorously to its technical dimension. The Jupyter notebook resources presented in this paper provide a gentle approach for doing so.

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

REFERENCES

Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In J. Goldstein, A. Lavie, C.-Y. Lin & C. Voss (Eds.), *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65–72). Association for Computational Linguistics. https://aclanthology.org/W05-0909/

Barba, L. A., Barker, L. J., Blank, D. S., Brown, J., Downey, A. B., George, T., Heagy, L. J., Mandli, K. T., Moore, j. K., Lippert, D., Niemeyer, K. E., Watkins, R. R., West, R. H., Wickes, E. Willing, C., & Zingale, M. (2019). *Teaching and learning with Jupyter*. https://jupyter4edu.github.io/jupyter-edu-book/

Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In K. Inui, J. Jiang, V. Ng & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3615–3620). Association for Computational Linguistics. https://aclanthology.org/D19-1371/

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the Natural Language Toolkit*. O'Reilly. https://www.nltk.org/book/

Buj, D. M., Ibáñez García, D., Parcheta, Z., & Casacuberta, F. (2020). NICE: Neural integrated custom engines. In A. Martins, H. Moniz, S. Fumega, B. Martins, F. Batista, L. Coheur, C. Parra, I. Trancoso, M. Turchi, A. Bisazza, J. Moorkens, A. Guerberof, M. Nurminen, L. Marg & M. L. Forcada (Eds.), *Proceedings of the 22nd annual conference of the European Association for Machine Translation* (pp. 329–338). European Association for Machine Translation. https://aclanthology.org/2020.eamt-1.35/

Christensen, T. P., Flanagan, M., & Schjoldager, A. (2017). Mapping translation technology

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

research in translation studies. An introduction to the thematic section. *Hermes – Journal of Language and Communication in Business* (56), 7–20. https://tidsskrift.dk/her/article/view/97199

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north American chapter of the Association for Computational Linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://aclanthology.org/N19-1423.pdf

EMT (2017). European Master's in Translation competence framework 2017. Website of the DG Translation of the European Commission. https://ec.europa.eu/info/sites/default/files/emt_competence_fwk_2017_en_web.pdf

Google (n. d.). Colaboratory. Frequently asked questions.

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

https://research.google.com/colaboratory/faq.html

Jia, Y, Carl, M., & Wang, X. (2019). How does the post-editing of neural machine translation compare with from-scratch translation? A product and process study. *Journal of Specialised Translation* 31, 60–86. https://www.jostrans.org/issue31/art_jia.pdf

Jupyter Team (2015). Jupyter notebook user documentation. https://jupyter-notebook.readthedocs.io/en/stable/index.html

Kaggle (n. d.). How to use Kaggle. Help and documentation. https://www.kaggle.com/docs/notebooks

Kenny, D., & Doherty, S. (2014). Statistical machine translation in the translation curriculum: overcoming obstacles and empowering translators. *The Interpreter and Translator Trainer* 8(2), 276–294.

Klein, G., Hernandez, F., Nguyen, V., & Senellart, J. (2020). The OpenNMT neural machine translation toolkit: 2020 Edition. In M. Denkowski & C. Federmann (Eds.),

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

Proceedings of the 14th conference of the Association for Machine Translation in the Americas (volume 1: research track) (pp. 102–109). Association for Machine Translation in the Americas. https://aclanthology.org/2020.amta-research.9/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv*. https://arxiv.org/abs/1301.3781

O'Brien, S., & Ehrensberger-Dow, M. (2020). MT literacy—a cognitive view. *Translation, Cognition & Behaviour* 3 (2), 145–164.

OpenNMT (2017). OpenNMT-py documentation. https://opennmt.net/OpenNMT-py/index.html

Open Source Initiative. (n. d.). The MIT license. https://opensource.org/licenses/MIT

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak & D. Lin (Eds.), *Proceedings of the 40th annual meeting of*

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

the Association for Computational Linguistics (pp. 311–318). Association for Computational Linguistics. https://www.aclweb.org/anthology/P02-1040/.

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, C. Hokamp, M. Huck, V. Logacheva & P. Pechina (Eds.), *Proceedings of the tenth workshop on statistical machine translation* (pp. 392–395). Association for Computational Linguistics. https://aclanthology.org/W15-3049/

Project Jupyter, Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Holdgraf, C., Kelley, K. Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., & Willing, C. (2019). Binder 2.0 – reproducible, interactive, sharable environments for science at scale. In F. Akici, D. Lippa, D. Niederhut & M. Pacer (Eds.), *Proceedings of the 17th Python in science conference (SciPy 2018)* (pp. 113–120).

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

https://conference.scipy.org/proceedings/scipy2018/pdfs/project_jupyter.pdf

Rei, R., Stewart, C., Farinha, A. C., & Lavie, A. (2020). COMET: A neural framework for MT evaluation. In B. Webber, T. Cohn, Y. He & Y. Liu (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2685–2702). Association for Computational Linguistics. https://aclanthology.org/2020.emnlp-main.213/

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th conference of the Association for Machine Translation in the Americas* (pp. 223–231). The Association for Machine Translation in the Americas. https://aclanthology.org/2006.amta-papers.25/

Tanaka, H., & Vanroy, B. (2019). Pyter3 0.3. Simple library to evaluate the translation edit rate. https://pypi.org/project/pyter3/

Krüger, R. (2021). An online repository of Python resources for teaching machine translation to translation students. *Current Trends in Translation Teaching and Learning E*, 8, 4 –20. DOI 10.51287/cttl_e_2021_2_ralph_kruger.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30 (NIPS 2017): Annual conference on neural information processing systems 2017* (pp. 1–11). https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating text generation with BERT. *arXiv*. https://arxiv.org/abs/1904.09675