Mimicking Neural Machine Translation History for Pedagogic Reasons

Vincent Vandeghinste
KU Leuven, Brussels, Belgium
Instituut voor de Nederlandse Taal, Leiden, the Netherlands
vincent@ccl.kuleuven.be

Abstract

In this paper we describe how we mimic the different steps in the historical development of NMT systems, all trained and evaluated on the same small data set. We do this for pedagogic reasons so students can see the effect of each of the steps on metrics like BLEU but also on qualitative examples, which the training scripts generate after each epoch. As MT paradigms, we discuss NMT training from scratch, fine-tuning pre-trained encoder-decoder models, and finally prompt engineering for decoder-only models. All models run in Kaggle sessions and all Python scripts and Jupyter notebooks are made available to the MT teaching community through GitHub and public Kaggle sessions.

1 Introduction

Neural machine translation (NMT) has undergone rapid architectural evolution during the past decade. Early sequence-to-sequence models based on recurrent neural networks (RNNs) were quickly followed by attention mechanisms, subword modeling, transformer architectures, multilingual models, pre-trained models, and more recently decoder-only large language models.

For translation students encountering machine translation (MT) for the first time, this rapid development can make the field difficult to understand. Modern state-of-the-art systems rely on highly complex pretrained architectures whose design motivations are not immediately obvious. Introducing such systems directly may therefore obscure the conceptual foundations of neural machine translation.

In this paper, we describe a pedagogical approach that addresses this challenge by mimicking the historical development of NMT. Rather than presenting current architectures directly, students learn to reproduce simplified versions of the models that marked major milestones in the development of the field, while also getting used to the MT experimental research paradigm.

This approach is implemented as the main part in the course *Machine Translation Advanced Topics* in the Translation Technology post-graduate programme for translators at KU Leuven. There are 27 contact hours and this counts for 3 ECTS points, so expecting a total average work load for students of 90 hours, including contact hours.

Until recently, we had been using OpenNMT (Klein et al., 2017), an open-source NMT platform that allowed easy training of NMT systems without much programming or other technical knowledge, making it an ideal tool to use in classes. But as development to OpenNMT was discontinued, it became very difficult to reconstruct the NMT history.

This is the main reason why we have developed a number of Python scripts with lots of commandline switches that mimic some of the functionality of OpenNMT. We have also created a series of Google Colab / Kaggle sessions in which these scripts are used.[1] Everything is made available via GitHub.[2]

[1] We use Kaggle as an alternative to the often used Google Colab environment, as Kaggle allows running Python notebooks while the user is offline and it has a fixed GPU quotum of currently 30 hours of GPU time per week.

[2] https://github.com/VincentCCL/MTAT/

In what follows, we show through evaluation with SacreBLEU (Post, 2018) (which was built-in in the scripts) and example translations how translation quality improves over the different inventions that have led to the current state-of-the-art in MT.

Section 2 discusses the dataset that was used. Section 3 discusses NMT training from scratch. Section 4 discusses building a multilingual NMT system and demonstrates a zero-shot system. Section 5 discusses fine-tuning and translation with pre-trained encoder-decoder models. Section 6 discusses prompt engineering with decoder-only models. And finally, Section 7 draws conclusions.

2 The dataset

All the Kaggle sessions make use of a dataset that was downloaded from OPUS (Tiedemann, 2012). We chose the Tatoeba (Tiedemann, 2020) English-Dutch dataset as it is rather small, in order to avoid long waiting times during class. Table 1 shows details about the dataset.

Tatoeba v2023-04-12 Properties	
nr of parallel sentences	79,541
nr of English tokens	579,984
nr of Dutch tokens	586,024

Table 1: Tatoeba v2023-04-12 properties

These data have been preprocessed similar as what is described in DataLit^{MT}'s (Hackenbuchner and Krüger, 2023) section on Data Preprocessing: entries with missing sentences are dropped, as well as duplicates and pairs where source and target sentence are equal, leaving us with 79,482 sentences, removing 59 sentence pairs (0.07%).

The next step consists of removing sentence pairs where either of the sentences contains more than 50 spaces, and those where the ratio between the source sentence number of words and the target sentence number of words (or vice versa) exceeds 2.5, removing another 101 sentences (another 0.12% of the initial set), leaving us with 79,381 sentences.

Then the data was tokenized with the SacreMoses tokenizer[3] and split into a training set, a development set and a test set. We limited the size of the development and test sets to 1000 parallel sentences each, again in order to not have to wait too long when running evaluations in a classroom setting.

[3] `https://pypi.org/project/sacremoses/`

In the remainder of this paper, we will not use or report about the test set, as this is kept as unseen data for students to perform a final evaluation once the best-performing systems have been selected based on the development set. Nevertheless, the cleaned data set with splits is made publicly available as a Kaggle dataset.

After each epoch, the training scripts show, depending on the options, the development set BLEU scores (Papineni et al., 2002) and a number of example sentences from the development set, in order to show students how results evolve during training, enabling them to see qualitative progress beside quantitative progress.

Table 2 shows 5 lowercased example sentences from the development set, with their lowercased reference translations, for which translation hypotheses are shown after each epoch.

Nr	Src/Ref	**Sentence** (lower cased)
1	SRC	nobody reads about my country .
	REF	niemand leest over mijn land .
2	SRC	i sleep in my car .
	REF	ik slaap in mijn auto .
3	SRC	there 're clean sheets under the bed .
	REF	er liggen schone lakens onder het bed .
4	SRC	i have a donkey .
	REF	ik heb een ezel .
5	SRC	betty drives fast .
	REF	betty rijdt snel .

Table 2: 5 example sentences from the development set

3 MT Models trained from scratch

We first discuss basic recurrent neural network models in Section 3.1, add subwording in Section 3.2, move to transformers in Section 3.3 and then present results on all the discussed models in Section 3.4.

3.1 Basic RNN models

After some explanation on how neural networks, word embeddings and plain RNN language modeling work, we start with training the RNN baseline, that is, a plain vanilla RNN encoder-decoder model (Cho et al., 2014; Sutskever et al., 2014). Table 3 shows the default hyperparameters for the RNN training script in the RNN column.

After the baseline, we have accordingly added gating to the RNN cells, comparing performance on Long Short-term Memory cells (LSTMs) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014), added

Setting	RNN	Transformer
Epochs	10	10
Batch size	32	32
Embedding size	64	512
Hidden size	128	2048
Encoder layers	1	6
Decoder layers	1	6
Number of heads	-	8
RNN type	RNN	-
Attention	none	self
Bidirectional encoder	no	-
Max sentence length	20	128
Learning rate	0.001	0.0005
Teacher forcing	0.7	1.0
Gradient clipping	1.0	1.0
Optimizer	Adam	AdamW
Lowercasing	yes	no
Subwording	none	SentPiece
Max source vocab	unlimited	8000
Max target vocab	unlimited	8000
Seed	42	42

Table 3: Default hyperparameters for the RNN and Transformer scripts. Some of these hyperparameters have been explained to the students, but for many this would lead too far as it would involve a technicality in maths which is not expected from translators. We provide them here mainly for reproducability.

bidirectionality to LSTM (*biLSTM*), and added cross-lingual attention (*att*) (Luong et al., 2015).

3.2 Subwording with SentencePiece

We also test the effect of applying SentencePiece subword tokenization (Kudo and Richardson, 2018) with unigrams, in two variants: with separate sentence piece models for source and target languages (see +*sp* in Figure 1 and Table 4) and with a single joint model for source and target language (see +*spj*). We always set vocabulary size at 8000.

3.3 Transformer models

For transformers we made a script that uses the `transformers` library from Hugging Face. We ran two versions of the Transformer model, a small version using as much as possible the hyperparameters from the RNN network (i.e., changing the default settings from the Transformer column in Table 3 for embedding size to 64, hidden size to 128 and maximum generated length to 20. Then we also trained a larger transformer model with the settings given in the Transformer column in Table 3. Note that for Transformers we have used separate sentence piece models for source and target language. Testing the accuracy of the model with a joint sentence piece model is left as an exercise for the students.

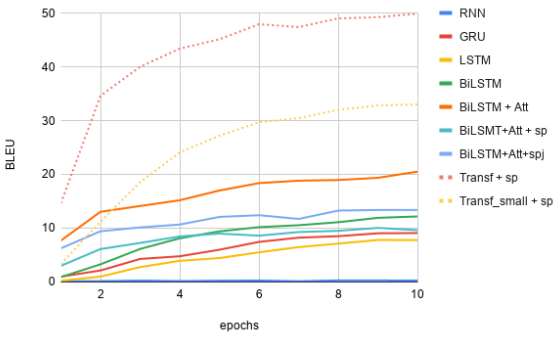

Figure 1: BLEU scores for the different models that were trained from scratch, over 10 epochs

3.4 Results

Figure 1 presents the learning curves for BLEU scores on the development set, clearly showing improvements with each newly introduced step. The difference betweeen LSTM and GRU is rather neglegible.

Table 4 shows the hypotheses of the respective MT models after 10 epochs. As we can see while vanilla RNNs only produce some high-frequency words, later models already may get some words right. As soon as we start introducing bidirectionality, we see output increasingly resembling full sentences. When attention is added results are even better, but still far from decent translations.

As Figure 1 shows, using sub-wording (*sp* and *spj* conditions) in our case does not improve further BLEU scores, but there is clearly a higher BLEU score for the joint model over the separate models.

Both transformer models score substantially better than the RNN models, and the larger transformer model clearly outperforms the small transformer. The improved performance is also visible from the generated examples, where the small transformer already shows some correctly translated sentences, and for the larger transformer we see only a single mistake in the five sentences.

4 Multilingual NMT (MNMT) models: testing zero-shot translation

We also want to demonstrate how multilingual models work. We take the French to Dutch Tatoeba set as an additional dataset, which was processed in the same way as was done in Section 2.

We created a function that takes a source file, a target tag and an output filename as its arguments and we use it on the files as indicated in Table 5.

Model	Nr	Hypotheses
RNN	1	is een het . .
	2	is een het . .
	3	is niet het van een . .
	4	is is . .
	5	is is . .
GRU	1	lijkt dat mijn mijn land . .
	2	heb in mijn auto . .
	3	zijn onder bed de bed .
	4	ben een . .
	5	rijdt snel . .
LSTM	1	leest op mijn land .
	2	ben in mijn mijn .
	3	heeft onder onder die niet . .
	4	heb een groot .
	5	rijdt snel .
BiLSTM	1	verdedigde over land land .
	2	slaap in mijn auto .
	3	zijn veel onder bed de bed .
	4	heb een . .
	5	rijdt beter .
BiLSTM	1	leest op mijn land .
+ att	2	slaap in mijn auto .
	3	zijn schoon op onder bed bed .
	4	heb een ezel .
	5	rijdt snel .
BiLSTM	1	las over mijn land .
+ att	2	ik slaapt in mijn auto .
+ sp	3	hang zijn bed onder bed onder bed .
	4	ik heb een . .
	5	t rijdt snel .
BiLSTM	1	leest van mijn land .
+ att	2	ik slaap in mijn auto
+ spj	3	zijnnt laket onder onder bed bed
	4	ik heb een ezel .
	5	tty rijd snel snel
Transformer	1	Niemand leest over mijn land .
small	2	Ik slaapt in mijn auto .
	3	Er zijn ketchup onder het bed .
	4	Ik heb een boot .
	5	Bet niet snel rijden .
Transformer	1	Niemand leest over mijn land .
	2	Ik slaap in mijn auto .
	3	Er zit schone lakens onder het bed .
	4	Ik heb een ezel .
	5	Betty rijdt snel .

Table 4: MT from scratch examples after 10 epochs. The **Nr** column refers to the example Nr in Table 2.

Original File	Tag	Ouput File
tatoeba-en-nl/train.en	<2nl>	en-nl-train.en_2nl
tatoeba-en-nl/dev.en	<2nl>	en-nl-dev.en_2nl
tatoeba-fr-nl/train.fr	<2nl>	fr-nl-train.fr_2nl
tatoeba-fr-nl/dev.fr	<2nl>	fr-nl-dev.fr_2nl
tatoeba-en-nl/train.nl	<2en>	en-nl-train.nl_2en
tatoeba-en-nl/dev.nl	<2en>	en-nl-dev.nl_2en
tatoeba-fr-nl/train.nl	<2fr>	fr-nl-train.nl_2fr
tatoeba-fr-nl/dev.nl	<2fr>	fr-nl.dev.nl_2fr

Table 5: Adding target language tags for multilingual MT

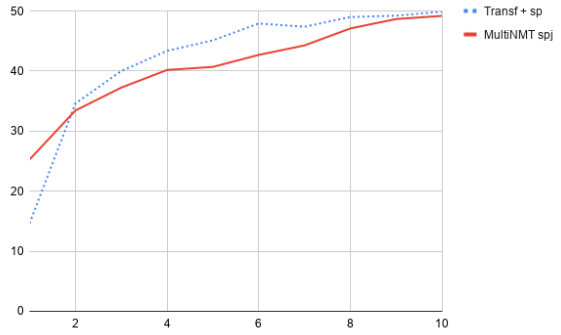

Figure 2: BLEU score for the multilingual NMT model. For ease of reference, the best-scoring system from Figure 1 is shown in a dotted line. Note that the two are not fully comparable, as the NMNT system is evaluated on a multilingual NMT file, not only on EN-NL.

Model	Nr	Hypotheses
MNMT	1	Niemand laira dans mon pays.
	2	Il y a du courant dans mon auto.
	3	Il y a des pteaux dans le lit.
	4	Il y a du cou.
	5	Betty rijdt snel.

Table 6: Translation hypotheses of the five example sentences in the zero-shot EN-FR multilingual setting

We create four datasets, in which the source sentences are prefixed with a target language tag. All training data was concatenated and a joint SentencePiece model was built. Then a transformer was trained with default settings (cf. Table 3) on these data. Results are presented in Figure 2, although the BLEU score is not fully comparable as the evaluation was done on the concatenation of the four development tests (See table 5).

Next, we tested the zero-shot capabilities of our model by translating the English development set in French. Table 6 shows example translations, from which we can see that the translations are not very good, as they contain a mixture of French and Dutch. We cannot evaluate this setting with metrics as we do not have the reference translations from English to French for these sentences.

5 Fine-tuning and using pre-trained Encoder-Decoders

For fine-tuning pre-trained encoder-decoders we have made separate scripts for T5 and mBART. For translation without fine-tuning, we also add M2M, NLLB and MADLAD-400.

Model	Nr	Hypotheses
T5	1	Niemand leest over mijn land.
	2	Ik slaapt in mijn auto.
	3	Er zijn snel bleeks op het bed.
	4	Ik heb een ees.
	5	Betty rijdt snel.
mT5	1	Niemand leest over mijn land .
	2	Ik slaap in mijn auto .
	3	Er zijn schoon kussens onder het bed .
	4	Ik heb een doek .
	5	Betty rijdt snel .
mBART (9 epochs)	1	Niemand leest over mijn land .
	2	Ik slaap in mijn auto .
	3	Onder het bed liggen schone lakens .
	4	< > Ik heb een ezel .
	5	Betty rijdt snel .
M2M (no finetuning)	1	Niemand leest over mijn land.
	2	Ik slapen in mijn auto.
	3	Er zijn schoon bladeren onder het bed.
	4	Ik heb een donkey.
	5	Betty rijdt snel.
NLLB (no finetuning)	1	Niemand leest over mijn land .
	2	Ik slaap in mijn auto .
	3	Er zijn schone lakens onder het bed .
	4	Ik heb een ezel .
	5	Betty rijdt snel .

Table 7: Encoder-Decoder examples after 10 epochs

5.1 T5-variants (Text-to-Text Transfer Transformer)

T5 (Raffel et al., 2020) was one of the first large-scale models to systematically explore how transfer learning can be unified across many NLP tasks within a single encoder-decoder architecture.

Variant mT5 (multilingual T5) (Xue et al., 2021) extends the T5 framework to the multilingual setting. Like T5, it treats all tasks as text-to-text generation, but it does so using a model pretrained on many languages rather than mainly English.

5.2 mBART (Multilingual Bidirectional and Auto-Regressive Transformer)

mBART (Liu et al., 2020) extends BART to the multilingual setting and was one of the first models to show that multilingual denoising pretraining alone can substantially improve machine translation.

Fine-tuning it within Kaggle is possible, but our run was stopped after 9 epochs because it exceeded the max allowed execution duration.

5.3 M2M (Many-to-Many)

Unlike T5, mT5, and mBART, which are primarily pretrained using monolingual denoising objectives, M2M-100 (Fan et al., 2021) is trained directly on large-scale parallel translation data. This makes it fundamentally a multilingual translation model rather than a general-purpose text generation model adapted later to MT.

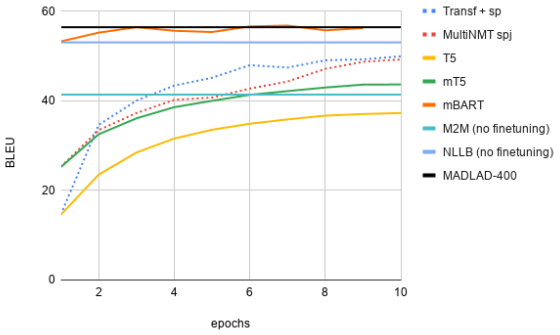

Figure 3: BLEU scores for the different pretrained encoder-decoder models over 10 epochs. The dotted line is the best scoring model as shown in Figure 1.

The M2M-100 models are too large to be fine-tuned comfortably in a standard Kaggle setup, so in this course we use the pretrained model directly for translation.

5.4 NLLB (No Language Left Behind)

The NLLB project (NLLB Team et al., 2022) extends massively multilingual machine translation to 200+ languages, with particular attention to low-resource and underrepresented languages. Its main goal is not only to scale translation to more languages, but also to improve quality for language communities that have historically received little support in MT.

Similar as for M2M, we do not fine-tune the model in Kaggle, but we can use the pretrained model directly for translation.

5.5 MADLAD-400 (Multilingual And Document-Level Large Audited Dataset)

MADLAD-400 (Kudugunta et al., 2023) represents a recent development in pretrained multilingual models that combines ideas from both encoder–decoder MT systems and text-to-text models such as T5.

Like for T5 and mT5, all tasks, including translation, are expressed as transformations from input text to output text. However, unlike earlier text-to-text models, it is trained explicitly for **massively multilingual translation**, covering hundreds of languages.

Similar as for M2M and NLLB, we do not fine-tune the model in Kaggle, but we can use the pretrained model directly for translation.

5.6 Results

As we can see in Figure 3, the T5 model scores a little better than the small transformer variant trained from scratch. The mT5 model scores a lot better, but still not as good as the large transformer model.

For mBART, we do not see a lot of improvement in fine-tuning, but nevertheless it is the best scoring model overall.

While we could not fine-tune M2M, NLLB nor MADLAD-400, due to hardware limitations in Kaggle, we show their performance in Figure 3 as straight lines, without fine-tuning. This shows that, even while M2M scores a BLEU score of 41, this is largely outperformed by NLLB, with a score of 53, which comes very near to the fine-tuned mBART score, so we can hypothesize that fine-tuning NLLB may beat mBART. MADLAD-400 scores 56.37 from the start, without fine-tuning, which is better than any of the fine-tuned systems.

Nevertheless, it should be noted that for these pretrained models data leakage between our development set and the data used for pretraining is rather likely, as the Tatoeba dataset is freely available so there is no reason why these systems would not have been trained on these data.

6 Prompt engineering with decoder-only models

We also introduce the students to prompt engineering with decoder-only models, and provide Kaggle sessions for working with GPT-2 (Radford et al., 2019), LLaMA (Touvron et al., 2023) and Mistral (Jiang et al., 2023), and we distinguish between different types of prompts, as shown in Table 8, such as, (1) zero-shot baseline, where we begin with a minimal instruction prompt; (2) format prompting – we make the structure of the task more explicit; (3) few-shot prompting – we provide one or more examples in the prompt; and (4) chat-style prompting. We also explore the effect of sampling and beam size.

Due to the limitations of Kaggle, we cannot download real size state-of-the-art models into the Kaggle environment. Therefore, for LLaMA, we also provide a Kaggle session with an API to the larger model running on a Hugging Face endpoint.

We also provide a Kaggle session which is using endpoints, such as Blablador (Strube, 2024) and Groq (Groq Inc., 2024), two services which offer access and compute for many different models.

Output of these decoder-only models shows that it is rather difficult and ad-hoc to make them output only the target sentence and nothing else, as they tend to overgenerate text. As such, overgeneration would largely affect the BLEU scores (not included here), but for the larger models, the output often contains nearly correct translations.

7 Conclusions

We have presented a number of scripts and notebooks that allow showing the impact of specific factors on NMT quality. A limitation of what we presented is that we have only trained / fine-tuned systems on a small dataset for a single language, but with these notebooks, it should be relatively easy for students in machine translation to train their own systems and test the effects of certain hyper-parameters and system architectures on the translation quality of their preferred language pair and domain.

While in this paper we have only used BLEU as an evaluation metric, calculated with SacreBleu, students have been taught how to use other metrics, be it through the use of MATEO (Vanroy et al., 2023) or through the use of scripts that perform evaluations with BERTScore (Zhang et al., 2020), BLEURT (Sellam et al., 2020) or COMET (Rei et al., 2020).

All the notebooks presented here should run in Kaggle, within the hard limitations set by the free version of Kaggle on GPU usage, processing time, disk space and RAM, these factors play. Nevertheless, if other GPU environments are available, training may go faster, fine-tuning of M2M and NLLB may become possible and working with larger open-weight decoder-only models may become feasible. Moreover, for the decoder models we have provided sessions that use free (for researchers) inference end-points for recent very large language models.

By sharing the scripts and making the notebooks public we hope to provide other teachers in MT to non-technical people with the right tooling to make students grasp the effect of the selection of training data.

It must be said that initially this course is very far outside the comfort zone of the students, of which only a minority has followed the first term course *Introduction to Machine Learning with Python*, but gradually most students get into the topic, as a lot of time is spent on making things

Prompt style	Example prompt
Zero-shot	Translate the following text from {source_lang} to {target_lang}. Only output the translation.\n\n{text}
Formatted	English: {text}\nDutch:
Few-shot	English: I eat apples.\nDutch: Ik eet appels.\n\nEnglish: She is at home.\nDutch: Zij is thuis.\n\nEnglish: {text}\nDutch:
Chat	You are a careful machine translation system. Translate faithfully and output only the translation. Translate this sentence from {source_lang} to {target_lang}. Only output the translation.\n\n{text}

Table 8: Example prompt styles and prompts

work on their own dataset. This involves quite a lot of individual help from the teacher, so the approach is probably only feasible for small groups. Nevertheless, during the course, students have some success experiences with the mechanics of MT training, and gradually they also see the MT quality improve. A downside is that training these models takes quite some time, which requires careful planning of when to explain MT theory versus when to do the hands-on work. All in all, most students that keep up the lessons till the end get a good grip on MT experimental methodology, the importance of data and finetuning and the general mechanisms of current-day language modeling.

Future work involves integrating the scripts as much as possible so it becomes even easier for students to play around with the hyper-parameters. We also want to investigate whether and how it would be possible to run these notebooks at the high performance computing facilities of the university instead of depending on commercial parties who may change running conditions as they please.

References

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(1):4839–4886.

Groq Inc. 2024. Groq API documentation. `https://console.groq.com/`. Accessed: 2026-04-03.

Hackenbuchner, Janiça and Ralph Krüger. 2023. DataLitMT – Teaching data literacy in the context of machine translation literacy. In Nurminen, Mary, Judith Brenner, Maarit Koponen, Sirkku Latomaa, Mikhail Mikhailov, Frederike Schierl, Tharindu Ranasinghe, Eva Vanmassenhove, Sergi Alvarez Vidal, Nora Aranberri, Mara Nunziatini, Carla Parra Escartín, Mikel Forcada, Maja Popovic, Carolina Scarton, and Helena Moniz, editors, *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 285–293, Tampere. European Association for Machine Translation.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jiang, Albert Q., Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv*.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In Bansal, Mohit and Heng Ji, editors, *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver. Association for Computational Linguistics.

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 (EMNLP)*, pages 66–71, Brussels. Association for Computational Linguistics.

Kudugunta, Sneha, Isaac Caswell, Biao Zhang, Xavier Garcia, Derrick Xin, Aditya Kusupati, Romi Stella, Ankur Bapna, and Orhan Firat. 2023. MADLAD-400: A multilingual and document-level large audited dataset. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook. Curran Associates Inc.

Liu, Yinhan, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. In Johnson, Mark, Brian Roark, and Ani Nenkova, editors, *Transactions of the Association for Computational Linguistics, Volume 8*, pages 726–742, Cambridge. MIT Press.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In Isabelle, Pierre, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia. Association for Computational Linguistics.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels. Association for Computational Linguistics.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI Blog.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1):5485–5551.

Rei, Ricardo, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In Webber, Bonnie, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Sellam, Thibault, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In Jurafsky, Dan, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Strube, Alexandre. 2024. Helmholtz Blablador and the LLM models' ecosystem. EuroSciPy 2024, Szczecin.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS'14: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pages 3104–3112, Cambridge, MA. Curran Associates, Inc.

Tiedemann, Jörg. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul. European Language Resources Association (ELRA).

Tiedemann, Jörg. 2020. The Tatoeba translation challenge – Realistic data sets for low resource and multilingual MT. In Barrault, Loïc, Ondřej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Yvette Graham, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Makoto Morishita, Christof Monz, Masaaki Nagata, Toshiaki Nakazawa, and Matteo Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vanroy, Bram, Arda Tezcan, and Lieve Macken. 2023. MATEO: MAchine Translation Evaluation Online. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 389–393, Tampere. European Association for Machine Translation.

Xue, Linting, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR 2020)*.