Language Resources and Evaluation
https://doi.org/10.1007/s10579-023-09704-w

ORIGINAL PAPER

Democratizing neural machine translation with OPUS-MT

Jörg Tiedemann[1] · Mikko Aulamo[1] · Daria Bakshandaeva[1] · Michele Boggia[1] · Stig-Arne Grönroos[1,2] · Tommi Nieminen[1] · Alessandro Raganato[3] · Yves Scherrer[1,4] · Raúl Vázquez[1] · Sami Virpioja[1]

Accepted: 30 October 2023

Abstract
This paper presents the OPUS ecosystem with a focus on the development of open machine translation models and tools, and their integration into end-user applications, development platforms and professional workflows. We discuss our ongoing mission of increasing language coverage and translation quality, and also describe work on the development of modular translation models and speed-optimized compact solutions for real-time translation on regular desktops and small devices.

Keywords Neural machine translation · Parallel corpora · Computer-assisted translation · Open source

1 Introduction

Language technology carries a growing responsibility in a society that is increasingly dominated by digital communication channels. Machine translation (MT) plays a decisive role in cross-lingual information access and will continue to grow as a crucial component in our natural language processing (NLP) toolbox, enabling inclusiveness and equity among people with different cultural and linguistic backgrounds. All the major IT companies recognize the importance of MT and push significant efforts into the development of internal translation solutions with slogans like "no language left behind"[1] and similar initiatives.

However, leaving MT to commercial exploitation comes with severe risks related to data privacy, transparency and inclusivity. The mission of OPUS (Tiedemann & Nygaard, 2004) and OPUS-MT (Tiedemann & Thottingal, 2020) is to push open and transparent solutions supported by the research community without profit-oriented goals in mind. The starting point is OPUS, a growing collection of public parallel data sets providing the essential fuel for open data-driven MT. Training data with

[1] https://www.microsoft.com/en-us/research/blog/no-language-left-behind/ and https://ai.facebook.com/research/no-language-left-behind/

Extended author information available on the last page of the article

Published online: 13 December 2023

good language coverage is crucial for high-quality MT. OPUS-MT builds on that collection and provides public translation tools and MT solutions. The long tail of languages with limited NLP resources creates one of the largest challenges of modern language technology, and thus we aim at improved language coverage in OPUS and OPUS-MT. This paper describes the infrastructure that we are building to support our mission (see Fig. 1 for a high-level overview of the various components and their connections).

Recent years have seen a revolution in natural language processing due to the advances in deep learning and computing facilities that support training complex neural network architectures. This success was only possible thanks to the availability of open source frameworks and growing public data sets. Another cornerstone in modern NLP is the distribution of pre-trained models that can be reused and adjusted to new tasks. Transfer learning has been shown to be very effective for many downstream tasks since it avoids expensive training procedures and draws significant benefits from unsupervised pre-training on raw text. Surprisingly, little has been done until recently with respect to public translation models and most teams still develop translation engines from scratch, even the most basic ones. However, increasing energy consumption and awareness of the environmental impact of deep learning call for a better and more sustainable use of resources. We propose OPUS-MT as a major hub for pre-trained translation models along with other initiatives that distribute deployable NLP models.

The most competitive architectures for neural language and translation models also present the drawback of having an ever-increasing size. Creating and even deploying such models becomes an obstacle and can only be done by well equipped units with sufficient High Performance Computing (HPC) backbones. Thus they tend to stay in the hands of large corporations that provide MT as a service. Even when creating high-quality models efficient enough to be run locally on the end-user's device would be possible, the corporations rarely have any incentive to publish such models with a license that would permit this.

Considering the importance of translation support, it is unfavorable if MT stays in the hands of a few high-tech corporations. Providing competitive public translation models with permissive licenses is, therefore, essential to avoid monopolies and to bring MT to the devices of end users, researchers and application developers without any strings attached and (implicit) commercial exploitation of its users. Many every-day users may not be aware of the dangers of data leaks and exploitation of personal information when using so-called free online services that feed into commercial products and targeted advertisements. In the spirit of open data and open source in NLP, OPUS-MT tries to democratize machine translation taking away the dependence on profit-maximizing services and tools. In the following we describe the OPUS ecosystem and how it supports this goal.

In Sect. 2 we focus on data outlining the principles of OPUS and the tools connected to our collection, including the OPUS-API, OpusTools and OpusFilter toolkits. The section also presents the Tatoeba translation challenge. In Sect. 3, we provide details about the OPUS-MT framework, the training pipelines and the integration of MT models into various platforms such as Hugging Face, the European

Language Grid and other end-user applications, as well as the OPUS-CAT professional workflow integration. Thereafter, we discuss benchmarks and evaluation as an important component in MT development in Sect. 4. Finally, we discuss our current efforts in scaling up language coverage and optimizing translation models in terms of speed and applicability in Sect. 5 before summarizing related work in Sect. 6 and concluding the paper with some final remarks.

2 The open parallel corpus OPUS

OPUS[2] has been a major hub for parallel corpora for about 18 years (Tiedemann, 2009, 2012; Tiedemann & Nygaard, 2004). It serves aligned bitexts (i.e. bilingually aligned parallel texts) for a large number of languages and language pairs, providing publicly available data sets for machine translation from various domains and sources. Currently, the released data sets cover over 600 languages and additional regional language variants that are compiled into sentence-aligned bitexts for more than 40,000 language pairs. In total there are ca. 20 billion sentences and sentence fragments that correspond to 290 billion tokens in the entire collection. The released data sets amount to about 12 TB of compressed files. Note that each sentence can be aligned to many other languages depending on the language coverage in individual sub-corpora. The distribution over languages and language pairs is naturally skewed and follows a Zipfian curve (Zipf, 1932) with a long tail of language pairs with little data, whereas only a few languages cover the main part of the data set. However, there are over 300 language pairs that contain one million sentence pairs or more, providing a good base for high quality machine translation. Note, that the collection will be skewed toward certain domains and the use of the resulting models will be affected by the domain coverage for individual language pairs as well.

The native format for OPUS is a simple standalone XML format that allows to include additional markup depending on the original source. Sentences are marked with appropriate sentence boundary tags and alignments are stored as standoff annotation in XCES Align format. Figure 2 shows an example of the annotation. Aligning text files based on sentence IDs has the advantage of scaling to large massively multilingual corpora such as software localisation data sets or Bible translations. Any document can be aligned with many translations or adaptations without repeating the essential content. The general principle in OPUS is to align all available language pairs to cover every possible language pair that can be extracted from the parallel data. Even alternative alignments can be provided without modifying the original corpus files, and without any impact on other alignments. Standoff alignment can also easily be augmented with meta-information such as alignment probabilities or link type information. Those additional features can be used to filter data sets according to certain conditions (see Sect. 2.2).

For convenience, OPUS also provides common data formats that support a seamless integration into machine translation pipelines and downstream applications.

[2] https://opus.nlpl.eu

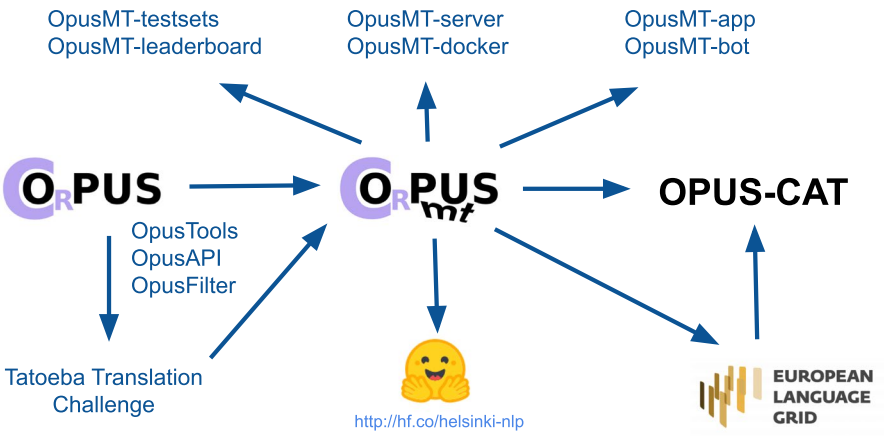

Fig. 1 OPUS-MT and its connections to other components, platforms and applications

Those files are automatically generated from the native XML format and released from the OPUS storage facilities. In particular, we provide plain text file corpora with aligned sentences on corresponding lines [referred as Moses-style bitexts due to their application in the Moses toolkit (Koehn et al., 2007)], and translation memory exchange files (TMX) that can be loaded from software mainly used by professional translators. Note that those derived files typically lose information such as additional markup, document boundaries as well as link type information and alignment scores. TMX files are also deduplicated (on the level of translation units, i.e. pairs of aligned sentences) and empty links are removed from both Moses and TMX file formats. The link counts reveal those discrepancies, which may result in significant differences for some sub-corpora in the collection.

For most corpora in OPUS, we also provide additional data files such as token frequency counts, word alignment files, extracted bilingual dictionaries and filtered phrase tables from statistical machine translation. We also provide monolingual data sets, some of which include data that goes far beyond the aligned texts. All of those released files are provided in a consistent file format and interface to make it convenient to integrate different subsets in systematic experiments and downstream applications. Note, however, that depending on the information available from the original sources there are slight differences in structures, markup and language IDs used in the various data collections. Notwithstanding, we try to follow common standards with mainly ISO-639-1 language codes with fallback to three-letter ISO-639-2 and extensions specifying regional variants when necessary.

Certain inconsistencies (for example in language labels) and other issues are unavoidable since all the data sets contain different levels of noise caused by file conversion, sentence boundary detection and automatic sentence alignment, but most of it stems from the original sources. With each import, we implement filtering and cleaning procedures, and we continually aim to increase the quality level of the data. New releases will provide the result of those efforts as well as filtering software that can be adjusted for individual needs (see also Sect. 2.3).

Basic corpus data with sentence boundaries (abbreviated for brevity):

```
<?xml version="1.0" encoding="utf-8"?>
<document>
  <CHAPTER ID="0">
    <P id="1"></P>
    <SPEAKER ID="1" LANGUAGE="DE" NAME="Rübig">
      <P id="2">
        <s id="1">Madam President, I saw a few boats landing at Parliament this ...</s>
        <s id="2">Not only were there language difficulties; the telephone line ...</s>
        <s id="3">I would be most obliged if the number on which the security ...</s>
      </P>
      <P id="3"></P>
    </SPEAKER>
```

Standoff annotation for sentence alignment (with scores):

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cesAlign PUBLIC "-//CES//DTD XML cesAlign//EN" "">
<cesAlign version="1.0">
 <linkGrp targType="s" toDoc="fr/2005/CES_AC71_2005_5SUMMARY.xml.gz"
                     fromDoc="en/2005/CES_AC71_2005_5SUMMARY.xml.gz">
  <link certainty="0" xtargets=";1 2" id="SL1" />
  <link certainty="0.612088" xtargets="1;3" id="SL2" />
  <link certainty="0.173077" xtargets="2;4" id="SL3" />
...
```

Fig. 2 An example of XML encoded OPUS data (corpus and sentence alignments). Sentence IDs are used to link sentences between two documents. Multiple sentences may appear in a link in which there sentence IDs are separated by spaces. Source and target language IDs are separated by a semicolon and empty links may also appear (like the first link in the example above where no source language sentence is aligned with sentences 1 and 2 in the target language)

Also note that OPUS so far focuses on sentence-level alignments in the tradition of sentence-level machine translation. Certainly, the trend goes towards increased context in any NLP application, MT included, due to the improved capabilities of modern architectures to cope with larger input and the ability to produce coherent output with long-distance dependencies across sentence boundaries. Moving the entire collection to paragraph- and document-level units is impossible as many data sets are made out of individual sentences and their translation correspondences out of any context. However, some resources already include structural information and document-level alignment and we plan to make such information more easily accessible to enable translation modelling on larger segments as well. However, so far, our efforts on machine translation are based on sentence-level models only as we will describe in more detail further down.

2.1 Finding data sets using the OPUS-API

The OPUS-API[3] is an online API for searching data sets within OPUS. Data sets can be filtered by corpus names, available source and target languages, the type of pre-processing and the released version of the corpus. The API responds with an output

[3] https://opus.nlpl.eu/opusapi/

in JSON format. The response contains information and simple statistics about the corpora, such as a download link to the necessary corpus files, size of the data files in bytes, the number of sentence pairs and tokens among other things. Figure 3 shows an example of the output.

In addition, the OPUS-API provides the functionality to list all available corpora and languages in OPUS, as well as more specific queries. For example, one can look up all available languages for a given corpus, or all target languages for a source language within all of OPUS or within a given corpus. These commands are useful for finding data sets for specific language pairs.

2.2 Fetching and processing parallel data with OpusTools

The OpusTools package[4] (Aulamo et al., 2020) enables easy access to the corpora and data files in OPUS. It provides a Python interface to the OPUS-API, methods for downloading and formatting the identified corpora, and some additional functionality for language detection and corpus splitting.

The main script in OpusTools is `opus_read` whose primary function is to read files from OPUS and to output the sentence pairs in various supported formats. Given source and target languages and a corpus name, `opus_read` downloads the necessary files via information from the OPUS-API. It parses XCES format alignment files and reads the corresponding sentences from compressed XML files to produce the requested output in the desired formats, including Moses-style bitexts and TMX. The `opus_read` script also provides the functionality to run basic data filtering. For example, one might only want to include one-to-one sentence alignment in the output. Additionally, it is possible to filter sentence pairs based on XML attributes such as sentence alignment confidence score (subject to the availability of the tags in the selected corpus). Furthermore, OpusTools contains the `opus_langid` script, which can be used for language identification of OPUS files using pycld2[5] and langid.py[6] (Lui & Baldwin, 2012). The script inserts language identification attribute tags in the XML files to make it possible to use language identification information for data filtering. In the future, we plan to systematically add metadata like automatically detected language tags to all data sets in OPUS. OpusTools will be useful for this task also for incoming data sets. While OpusTools only supports basic data filtering and conversion methods, the OpusFilter toolbox described below offers a much wider range of options for processing the corpora.

[4] https://github.com/Helsinki-NLP/OpusTools
[5] https://github.com/aboSamoor/pycld2
[6] https://github.com/saffsd/langid.py

```
{
  "corpora": [
    {
      "alignment_pairs": 29903792,
      "corpus": "OpenSubtitles",
      "documents": 38969,
      "id": 328013,
      "latest": "True",
      "preprocessing": "xml",
      "size": 297683,
      "source": "en",
      "source_tokens": 257890401,
      "target": "fi",
      "target_tokens": 161280297,
      "url": "https://object.pouta.csc.fi/OPUS-OpenSubtitles/v2018/xml/en-fi.xml.gz",
      "version": "v2018"
    },
    ...
  ]
}
```


Fig. 3 An example of OPUS-API's JSON output showing the first item when querying for the latest version of the English–Finnish OpenSubtitles corpus in XML format

2.3 Cleaning and preparing data sets with OpusFilter

OpusFilter[7] (Aulamo et al., 2020) is a toolbox for filtering and combining parallel corpora. The toolbox supports three main types of operations: corpus preparation tasks, essential preprocessing steps and, finally, various types of filters that aim to remove noise and unwanted content. In contrast to tools that implement a single filtering method (e.g. Sánchez-Cartagena et al., 2018; Xu & Koehn, 2017), OpusFilter allows to apply different approaches, including custom filters defined by the user. Moreover, it can be used for other corpus manipulation tasks needed when building MT models, for example joining corpora, splitting them into training and test sets, and applying tokenization.

OpusFilter can connect to the OPUS corpus collection via the OpusTools library, but can also operate on local text files to process any monolingual, bilingual, or multiparallel corpora. A YAML[8] configuration file defines the pipeline to transform raw corpus files to clean training and test set files. The same pipeline can be generalized over multiple language pairs. A single, easy-to-share configuration file design is central in our efforts to provide transparent solutions and to alleviate the reproducibility issues of experiments that affect the current MT research (Marie et al., 2021).

2.3.1 Corpus-level data preparation

The first steps in parallel corpus preparation are typically related to obtaining, joining and splitting the different corpora. The interface to the OPUS collection via OpusTools allows automatic downloading of the existing corpora. OpusFilter

[7] https://github.com/Helsinki-NLP/OpusFilter
[8] https://yaml.org/

supports many common text file operations on parallel files: concatenation, head (*N* first lines), tail (*N* last lines), and slice (lines from *N* to *M*). There is a command for taking a random subset of a specific size from a corpus and splitting data into two subsets with given proportion. Another command allows combining multiple translations for the same segments in separate files into a single parallel set containing all available combinations or a sampled subset of them.[9] Once corpus files have been created, one can proceed with essential text preprocessing described below.

2.3.2 Segment-level preprocessing

The preprocessing step can be used to apply a number of segment-level preprocessors to monolingual or parallel corpora. The implemented preprocessors include whitespace normalization, custom regular expression substitutions, monolingual sentence splitting,[10] tokenization,[11] Chinese word segmentation,[12] Japanese word segmentation,[13] and subword segmentation with BPE[14](Sennrich et al., 2016) or Morfessor[15](Virpioja et al., 2013). Custom preprocessors can be added by implementing a simple Python class for them.

2.3.3 Filtering

OpusFilter includes two types of filters. In the scope of a corpus, duplicate filtering provides ways to either remove identical segments (optionally ignoring casing and non-letter characters), either in all or just some of the languages, within a corpus. It is also possible to remove those segments from one corpus that overlap with another corpus (e.g. training data segments that overlap with the test data).

The rest of the filters work segment-by-segment, either accepting or removing each independently of the others. The filters currently implemented by OpusFilter include various length-based filters, language identification filters (based on `langid.py`[16] (Lui & Baldwin, 2012), `pycld2`,[17] and `fasttext`[18] (Joulin et al., 2016; Joulin et al. 2017), script, special character, and similarity filters, filters using varigram language models[19](Siivola et al., 2007), filters using word alignment models[20](Östling & Tiedemann, 2016), and sentence embedding filter that

[9] For example, three Spanish translations and three French translations for the same English document can be combined to have nine Spanish–French pairs per each segment in the document.
[10] https://github.com/mediacloud/sentence-splitter
[11] https://github.com/mingruimingrui/fast-mosestokenizer
[12] https://github.com/fxsjy/jieba
[13] https://github.com/SamuraiT/mecab-python3
[14] https://github.com/rsennrich/subword-nmt
[15] https://github.com/aalto-speech/morfessor
[16] https://github.com/saffsd/langid.py
[17] https://github.com/aboSamoor/pycld2
[18] https://fasttext.cc/
[19] https://github.com/vsiivola/variKN
[20] https://github.com/robertostling/eflomal

estimates sentence similarity based on multilingual LASER embeddings[21] (Artetxe & Schwenk, 2019).

The segment-by-segment filters that can be used in OpusFilter specify two methods: the `score` method determines a numerical or boolean output for the segments, and the `accept` method makes a decision whether to accept or reject the segments with the given score. The score may be just one value or a vector of multiple values. For example, `LengthFilter` returns the number of characters or words in the segment: one for monolingual corpus, two for bilingual corpus, and so on. The minimum and maximum length parameters determine whether the segment is accepted or not.

Instead of directly applying the filtering decisions, OpusFilter can output the scores from the filters. They can be used to inspect the data and determine reasonable thresholds for filtering. For this, OpusFilter includes tools for analyzing and visualizing the scores. In addition to direct filtering, the filter scores can also be used to train a classifier to make the filtering decision based on a combination of the scores (Vázquez et al,. 2019).

2.4 The Tatoeba translation challenge

Recently, we created a special compilation of data released under the label of the Tatoeba translation challenge,[22] (TTC for short) (Tiedemann, 2020) to overcome some of the complications present in models trained on OPUS data. OPUS is a dynamic data collection with occasional new dataset releases, hence there are plenty of ways for compiling data sets from the collection and the coverage may always change. Moreover, data preprocessing and cleaning greatly influences the quality of MT, and frequent duplicates across all subsets together with substantial noise may have a negative influence on learning procedures.

The TTC is named after the selected test and development data, which we take from the Tatoeba corpus, a dataset of user-contributed translations in hundreds of languages. The latest release of the TTC includes 32 billion translation units[23] in 4,024 bitexts covering 487 languages. We compile the training data from OPUS corpora and focus on creating a data set with consistent language labels that can easily be used for systematic machine translation experiments. All bitexts are deduplicated and filtered. We provide randomly shuffled training sets and dedicated development and test data to support a consistent experimental setup.

We divide the data set into several sub-challenges depending on the availability of training data. Details of the setup are available in Tiedemann (2020). The current state of our machine translation development with respect to the TTC is monitored using our dashboard (see Sect. 4.1). Geographic visualizations (see Fig. 4) also help

[21] https://github.com/yannvgn/laserembeddings

[22] https://github.com/Helsinki-NLP/Tatoeba-Challenge

[23] Typically, translation units refer to aligned sentence pairs but they also include units of multiple sentences aligned to corresponding translations as it is often the case in various parallel data sets. Tatoeba also frequently includes units with more than one sentence.

to see the gaps in language coverage, which we try to fill in OPUS-MT. We also complement the evaluation with results from other established benchmarks. This is necessary as user contributed data such as the TTC benchmark can have various issues and a systematic quality control is beyond our capabilities. Furthermore, TTC refers to a specific kind of content mostly targeted at language learners and travelers with a focus on every-day language and rather short and frequent expressions.

We aim at regularly updating the TTC dataset in order to capture the growing support for language pairs and domains. Releases of benchmarks will be updated more frequently, depending on the growth of the original Tatoeba database. We encourage the reader to contribute to the collection of translations[24] to support our project. Training data releases will be compiled as needed whenever large new collections enter the OPUS collection.

3 Open machine translation with OPUS-MT

With OPUS-MT, we strive to provide public state-of-the-art translation solutions and to be a major hub for pre-trained translation models. OPUS-MT is based on Marian, an efficient implementation of neural machine translation (NMT) in pure C++ and with minimal dependencies (Junczys-Dowmunt et al., 2018). Marian is a production-ready framework and includes optimized routines that enable a scalable approach to development and exploitation of modern MT systems. In more detail, OPUS-MT targets two main objectives:

1. *Training pipelines and models (Section 3.1)*: Scripts and procedures that can be used to systematically train neural MT models from public data collected in OPUS. The recipes include all necessary components that are required to create competitive translation models, including data preparation, model training, back-translation and other kinds of data augmentation, as well as fine-tuning and evaluation. Using the pipelines, models are trained on a large scale and released with permissive licenses to be reused and integrated in various platforms and workflows.
2. *MT servers and integration (Sections 3.3 and 3.4)*: Pre-trained models need to be integrated into various platforms to make them widely applicable for end-users, translation professionals, system developers and general MT researchers. OPUS-MT provides simple server applications that can be used to deploy translation engines. The project also emphasizes integration into external infrastructures such as the Hugging Face `transformers` library and model hub (Sect. 3.3.1) and the European Language Grid (Sect. 3.3.2). Professional workflows are supported through plugins and installation packages tailored towards end users.

[24] https://tatoeba.org/en/users/register

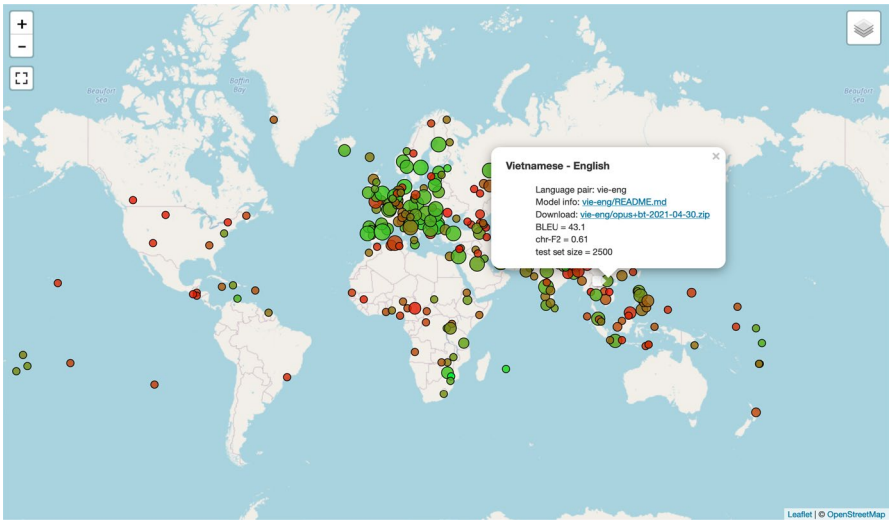

Fig. 4 Language coverage of translation models visualized on an interactive map. Geolocations of languages are taken from Glottolog and dot colors indicate the translation quality in terms of an automatic evaluation metric measured on the Tatoeba test set in this case on a scale from green (best) to red (worst). Smaller circles refer to smaller, less reliable test sets

The following sections provide an overview of the implementation and functionality of the training pipelines, the release procedures and documentation of pre-trained translation models, the implementation of server applications, and the integration into various platforms and software packages.

3.1 Training pipelines

The main purpose of OPUS-MT training pipelines is the integration of data preparation with flexible recipes for running experiments and massive NMT pre-training. Our intention is to create reusable procedures that we can run internally for creating the models we would like to use and release. However, we also share those recipes since we want to create a transparent approach that can be inspected and replicated.

The implementation of training pipelines is based on GNU Make[25] recipes. The `make` command and makefiles are well established work horses in the automation of compilation workflows and with that they provide a stable and tested environment, which is highly beneficial for our purposes. The philosophy of defining build targets that have various dependencies fits very well in the picture of model training and NLP experiments. Makefiles define the recipes and dependency chains that need to be followed in order to produce the final product, in our case NMT models and release packages.

[25] https://www.gnu.org/software/make/

There are several advantages that we benefit from when using makefiles:

- Dependencies are automatically resolved and the build process is interrupted if essential dependencies cannot be resolved.
- Intermediate files and results can be reused and do not have to be built again if not necessary. Timestamps automatically determine whether targets need to be re-built and updated, which may affect the entire pipeline.
- Targets can be built in parallel and the `make` command determines which recipes can run simultaneously without breaking dependencies defined by the internal recipes.
- Unnecessary intermediate files are deleted at the end of the process.
- Static pattern rules can be used to create template recipes, which can help to define generic procedures that apply to many related tasks.
- Many in-built functions support the manipulation of variables, file names and build workflows.

OPUS-MT tries to make use of those advantages and implements recipe chains that cover all necessary sub-tasks for training, testing and releasing NMT models. The goal is to provide various high-level `make` commands that can run in batch mode with variables that control properties and procedures to enable a systematic exploitation of massively parallel data sets. Without those generic recipes it would not have been possible to train such a large amount of models that we were able to release already (over 2,300 models at the time of writing, 744 multilingual ones among them with varying language coverage on source and target side).

Our training pipelines are stored in a public git repository[26] and build recipes are subject to change as we continuously develop the system. The main components and principles stay the same and documentation is provided within the repository. Here, we only provide a high-level overview of the package and refer the interested reader to the original source.

3.1.1 Setup and basic training

Setting up and starting training batch jobs is straightforward and is optimized for modern Linux-based systems with sufficient storage space, the availability of a CUDA compatible GPU and access to online sources. Installing the pipelines and software dependencies can be done on the command-line using the installation recipes:

```
git clone https://github.com/Helsinki-NLP/OPUS-MT-train.git
cd OPUS-MT-train
git submodule update --init --recursive --remote
make install
```

[26] https://github.com/Helsinki-NLP/OPUS-MT-train

Training generic models can be done by basic high-level targets that take care of the entire pipeline and sub-tasks necessary for fetching data and starting the training procedures. The documentation includes an extensive list of command line variables to control the properties and parameters of the various sub-tasks. However, the most essential variables to be set are the source and target language specifications:

```
make SRCLANGS="fi et" TRGLANGS="da sv en" data
make SRCLANGS="fi et" TRGLANGS="da sv en" train
make SRCLANGS="fi et" TRGLANGS="da sv en" eval
```

The above example demonstrates how easy it is to create a (multilingual) neural OPUS-MT model by just giving a set of language IDs to be included in the model (Finnish and Estonian as source languages and Danish, Swedish and English as target languages in this case). Note that the first command can be skipped because `data` is a prerequisite for the training recipe that will fetch and prepare data in any case.[27] The recipes are designed to take care of fetching all available data sets from OPUS with combinations of source and target languages. A random disjoint sample (in all language combinations) from a dedicated corpus will be used as validation and test data and remaining data goes into training. There is a manually specified hard-coded priority list of OPUS corpora that will be used as test and development data focusing on language coverage and relative cleanliness. The list and mechanisms are likely to change over time with further evolvements of OPUS and the OPUS-MT training recipes. Alternatively, the TTC data sets can be used to apply a consistent split into train, development and test data. Models that use multiple target languages will automatically use target language labels to augment the completely shared transformer model.

In general, OPUS-MT will use the latest releases of OPUS to prepare the data. Some basic sanity checking and filtering (removal of non-printable or broken Unicode characters and some length-based filtering) is done but no major changes are applied to the sources. Subword segmentation models are trained using SentencePiece (Kudo & Richardson, 2018) and word alignments are created using `eflomal` (Östling & Tiedemann, 2016) to feed the guided alignment feature that is used by default.[28] The default parameters provide a good fit for regular baseline models but we encourage further optimization of hyper-parameters. For more details on how to control the setup, please look at the documentation of the recipes themselves. The `eval` target will compute automatic evaluation metrics on heldout data. Another target (`eval-testsets`) can be used to also benchmark with other test sets available for supported language pairs. Those need to be available in the repository.

[27] Note that the `eval` target is not implemented with `train` as a pre-requisite in order to avoid that calls for evaluation automatically trigger expensive data fetching and training procedures.

[28] Neural word aligners like awesome aligner (Dou & Neubig, 2021) could be used as well but we prefer a language-agnostic aligner like `eflomal` that does not require pre-trained language models with a certain language coverage and the potential need of additional fine-tuning.

OPUS-MT also supports temperature-based data sampling (Arivazhagan et al., 2019), which is important to balance the availability of specific language pair examples in multilingual translation models. Adding a parameter to the training data creation recipes will influence the proportions used for skewed data sets. To give a practical example: adding `MAX_DATA_SIZE=1000000 DATA_SAMPLING_WEIGHT=0.2` to the `make` command will cause the system to select proportionally to the temperature-adjusted likelihood $p_l^{1/5}$ of observing an example in language l from the entire data set. The additional parameter `MAX_DATA_SIZE` sets the sample size to one million examples for the largest language pair and all other language pairs will be down- or over-sampled according to the sample rate.

3.1.2 Batch jobs on HPC infrastructure

The main mode of running OPUS-MT training pipelines is to use batch jobs on high-performance computing (HPC) clusters using the Slurm Workload Manager[29] as a job scheduler. To support a flexible use of all recipes, OPUS-MT implements generic pattern rules to turn any recipe (or makefile target) into a Slurm batch job. This is done by defining a special suffix that can be appended to any target available in the makefile-based build system. When adding such a suffix, the recipe will generate a Slurm script with the original target as a goal and submit it to the job scheduler using the setup specified for the HPC-specific environment. Note that those settings are now basically hard-coded to work in our own environment and adjustments need to be made to match the local installation of your HPC cluster. Below, you can see an example for submitting the training job for the same model discussed above with a walltime of 72 h (the `.submit` suffix triggers the creation of the batch job script and the submission to the Slurm engine):

```
make SRCLANGS="fi et" TRGLANGS="da sv en" WALLTIME=72 train.submit
```

Various variables can be used to control the behavior and resource allocation of those jobs. For details, have a look at the documentation and definitions in the source code.

3.1.3 Data augmentation

Data augmentation is particularly important for less resourced languages. However, even better resourced languages benefit from back-translation (Sennrich et al., 2016) and other techniques that increase the coverage of the data. OPUS-MT implements convenient procedures to produce back-translations and pivot-based triangulation data.

By default, monolingual data extracted from Wikipedia and other public data sets provided by the Wikimedia foundation are available for back-translation. For

[29] https://slurm.schedmd.com/overview.html

supported languages, OPUS-MT fetches the previously prepared data from our object storage and uses existing models in the opposite translation direction to translate them back into a source language. Back-translation is done for individual language pairs, but multilingual models can certainly be used besides of bilingual ones for that purpose, too.

Training OPUS-MT models in different directions can be done iteratively and, thus, back-translated data can improve step by step and with that the quality of translation models in both directions can go up (Hoang et al., 2018). This is especially useful in low-resource settings where initial back-translations may be quite noisy and translation quality first needs to reach reasonable levels in both directions. An example of such an iterative back-translation procedure is illustrated in Fig. 5, where a model for translating from English to Central Bikol is improved with several iterations of back-translation.

Another effective method for data augmentation is triangulation and pivot-based translation. Many data sets are English-centric, but the demand for direct translation between non-English languages is certainly growing and still under-explored. A practical real-world example arose in the on-going crisis in Ukraine. Refugees moving to various European countries need support to manage information flow and communication in the local languages. Figure 6 illustrates the various ways of augmenting data for improved translation, in this example between Ukrainian and Finnish (in both directions).

Triangulation of English-centric data is an easy way of producing additional training material. MultiParaCrawl[30] is created in this way by pivoting alignments on identical sentences in English. A non-negligible amount of the 1.5 million sentence pairs for Finnish–Ukrainian could be extracted in this way from the original English–Ukrainian and English–Finnish bitexts in ParaCrawl.

Pivot-based translation is another straightforward way of producing artificial training data. Translating one side of a bitext can turn existing data sets into a synthetic parallel corpus for a different language pair. This is especially useful if strong high-quality models can be used to perform that automatic translation. In our running example, we can use a multilingual model for related East Slavic languages to translate from Russian to Ukrainian and another optimized model for translating English into Finnish to transform English–Ukrainian and Russian–Finnish data into the desired Finnish–Ukrainian language pair. Note that we include both synthetic source language data (from the English–Ukrainian corpus where English is translated to Finnish) and synthetic target language data (from the Russian–Ukrainian corpus with Russian translated into Ukrainian). Typically, synthetic target language is not preferred because of the additional noise that may enter a translation model. However, given the quality that can be achieved for closely related languages, this procedure is stable enough and quite effective as another means of data augmentation.

Multilingual models would be an alternative for implicitly including pivot languages, but the pivot-based direct translation approach has the advantage that we

[30] https://opus.nlpl.eu/MultiParaCrawl.php

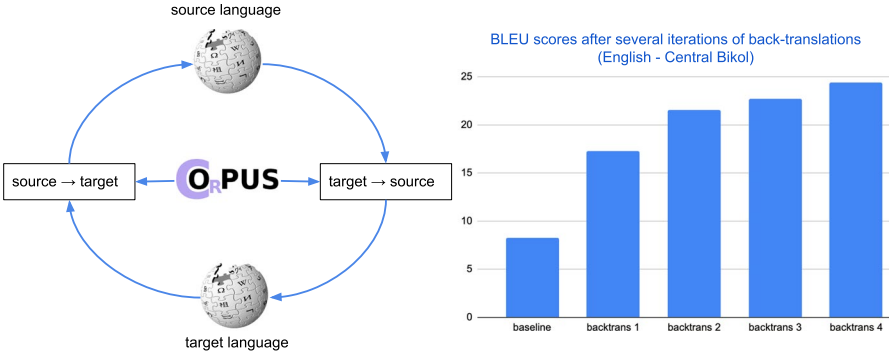

Fig. 5 Iterative back-translation as a means of data augmentation. OPUS-MT uses Wikipedia content as monolingual data and the example illustrates the iterative improvements (in the barchart to the right) for translations between English and Central Bikol (an Austronesian language spoken in the Philippines) in terms of BLEU scores. Monolingual data coming from Wikipedia is translated in several rounds of improved forward and backward translation using the fresh translations to create the current synthetic back-translated data for augmenting the translation models in both directions

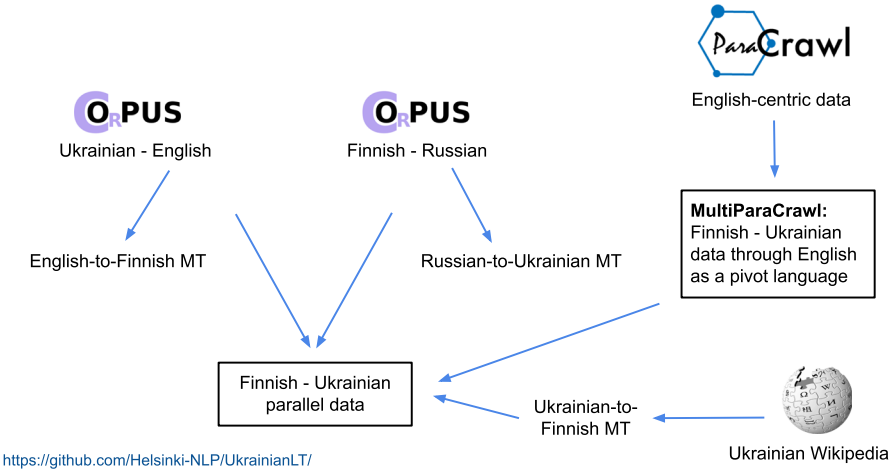

Fig. 6 Data augmentation using triangulation and pivot-based machine translation. Synthetic training data can be created from English-centric parallel corpora and automatic translations of some pivot language using existing OPUS-MT models. The illustration shows the approach on the example of Ukrainian–Finnish translation

do not need to increase the capacity of the translation model (by blowing up the number of parameters) to accommodate additional languages nor do we need to be concerned about balancing between different languages in mixed language data. Furthermore, we can once again benefit from existing pre-trained models that have already been optimized in various ways, avoiding to re-learn everything from scratch.

3.1.4 Fine-tuning

Usually, it is also common to fine-tune models for specific tasks or domains. OPUS-MT supports fine-tuning in a way that a short cycle of additional training based on small task-specific training data can be triggered. A common situation is that a local translation memory can serve as additional training material. Special recipes enable such a fine-tuning process.

Another common scenario is that multilingual models can be fine-tuned for specific language pairs. The idea is to take advantage of transfer learning in the general training phase and then to use language-specific data to fine-tune the model for that particular language pair. This is also supported by OPUS-MT and the released collection of training pipelines and recipes. However, in our experience, it is difficult to define proper learning parameters to avoid over-fitting and catastrophic forgetting. Therefore, we typically do not release such fine-tuned models and leave it up to end-users or developers of specific solutions to perform such a process.

3.1.5 Evaluating and releasing

The final task we want to discuss in this section refers to evaluation and packaging of models. The targets for testing the models have already been mentioned earlier. The OPUS-MT repository, furthermore, includes common benchmark test sets and those are useful to compare model quality with other established and published research. We are currently working on a more principled way to benchmark all released models. More information about that effort is given in Sect. 4.

An important feature is also the creation of release packages. Special targets are defined to collect all necessary information, generate documentation in terms of readme files and YAML files together with benchmark results and other essential details. Release packages contain models including essential pre- and post-processing scripts, subword segmentation models, and decoder configurations. Training log files are also included. For internal use, we also support convenient functions for storing and fetching models as well as intermediate work files from our external storage service. Various procedures are defined to update information available for external users including lists of released models and important properties that define their functionality.

Releases are done inside of ObjectStorage containers with a CEPH backend,[31] which are linked from project websites and GitHub markdown pages. Information about their performance is also available and more information on benchmarking will be given further down in Sect. 4.

[31] Ceph is an open-source, distributed storage system, see https://ceph.io/.

3.2 Machine translation server applications

OPUS-MT also provides ways of deploying models and building translation servers to demonstrate and use MT from web applications or through service APIs. Our implementations[32] leverage the efficient decoder released as part of Marian-NMT[33] in its server mode to create a running translation service. Two alternative applications are available, one based on the Tornado web framework and another one implementing websocket services that can easily be deployed on common Linux servers and virtual machines.

The Tornado-based solution is also containerized using docker and can in this way be deployed in various environments. Simple web frontends demonstrate the usage of the service APIs. Configuration and setup is straightforward using the released packages from OPUS-MT. The server backends take care of all basic pre- and post-processing (including sentence splitting and subword tokenization) and provide a simple JSON interface to the actual server. Both applications can also combine several translation services and provide them through the common interface and API. Models to be supported can be specified in configuration files. An example for a Tornado web app configuration is shown in Fig. 7.

The websocket service provides some additional functionalities that can be useful when deploying OPUS-MT models. There is an additional router daemon that can serve as a central access point, which is able to connect to various translation backends. The integration of multilingual models is also well supported. Furthermore, the output also provides the segmentation into subword units and an alignment between them, which is meaningful for models that are trained with the guided alignment feature of Marian-NMT.[34] Interfaces can then show links between input and output tokens to better trace the connections between source and target language inside of the translation model. Alignments can also be useful for additional post-processing features built on top of the translation service itself like placing tags and formatting information. An example output is shown in Fig. 8. Furthermore, the server backend also implements a cache that speeds up translation substantially when identical sentences are passed into the engine. The cache implements a simple yet efficient lookup function in a hashed database, which is stored on the local disk of the server.

Our web applications are mainly provided for demonstration purposes, but they can be used as a starting point for serious platform integration and real-world applications, as we will discuss in the next section.

[32] https://github.com/Helsinki-NLP/OPUS-MT

[33] https://marian-nmt.github.io/

[34] This feature allows to guide one of the cross-lingual attention heads in the transformer with pre-computed token alignments. In this way, we obtain an attention pattern that specializes on word alignment and it also helps to kickstart the training procedures with relevant prior knowledge, which can be useful especially in low-resource settings.

```
{
    "en": {
        "es": {
            "configuration": "./models/en-es/decoder.yml",
            "host": "localhost",
            "port": "10001"
        },
        "fi": {
            "configuration": "./models/en-fi/decoder.yml",
            "host": "localhost",
            "port": "10002"
        },
    }
}
```

Fig. 7 A simple configuration for an OPUS-MT server using the Tornado web app implementation providing services for English–Spanish and English–Finnish machine translation

```
{
    "alignment": [
        "0-0 1-1 2-2"
    ],
    "result": "Huomenta, Suomi",
    "segmentation": "spm",
    "server": "192.168.1.15:40002",
    "source": "sv",
    "source-segments": [
        "\u2581Godmorgon , \u2581Finland"
    ],
    "target": "fi",
    "target-segments": [
        "\u2581Huomenta , \u2581Suomi"
    ]
}
```

Fig. 8 A translation response from a websocket server using OPUS-MT to translate from Swedish to Finnish. The final result after post-processing is available in the “result" item and “alignment" refers to subword token alignments between source and target language segments. Source and target segments list subword tokens separated by whitespaces. The Unicode character `\u2581` at the beginning of some of them indicates the connection to the previous token (i.e. the need for removing the preceding space), which is commonly used by the SentencePiece tokenizer (Kudo, 2018)

3.3 Platform integration

The web services and applications described in the previous section already demonstrate the practical use of OPUS-MT models beyond the pure NLP research field. But even in research, platform integration becomes increasingly important: training is expensive and developing models from scratch becomes rarer in current approaches based on transfer learning. Integration into popular libraries and platforms is, therefore, essential to avoid unnecessary overheads of getting started with the basic facilities needed to make progress in research and development.

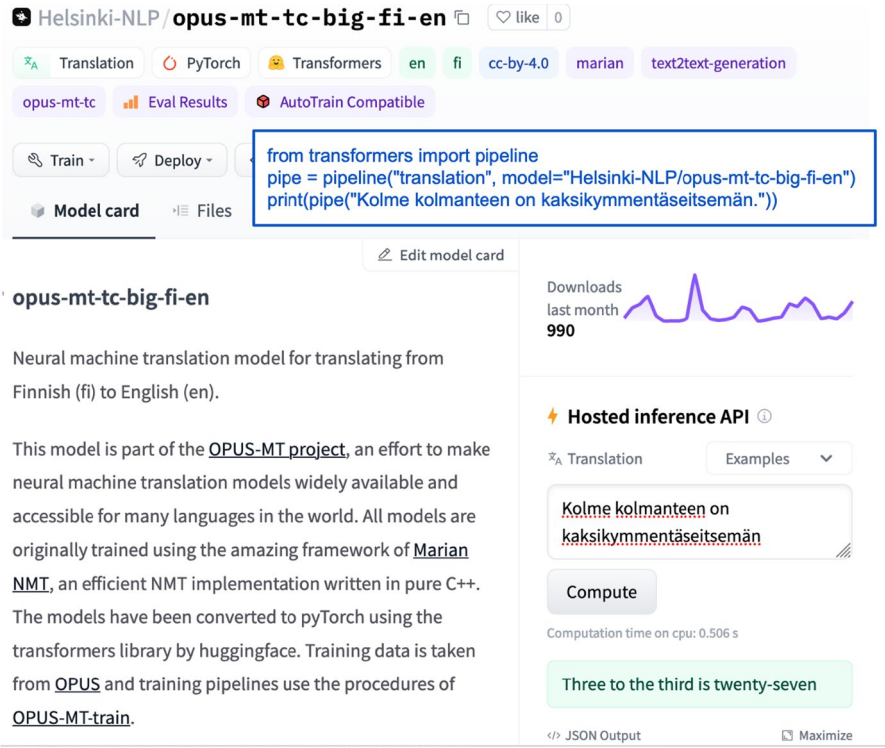

Fig. 9 An example of an OPUS-MT model card in the Hugging Face model hub. The blue text box on top of the screenshot shows a three-line code snippet for using the model from the `transformers` library. The model card includes information about the use of the model, supported languages and links to relevant project websites with further information about the original model and data sets used for creating it. The model hub also provides a live inference API that can be used to test the model (in the right column) and download statistics are also shown on the top of that column

In this section, we describe the efforts of connecting OPUS-MT to various external platforms and software packages in order to make our models widely available and accessible for end-users, application developers and basic NLP researchers. We only provide a few examples of possible use cases. Many other applications are potential platforms where OPUS-MT can be integrated. For example, our models are already available through Tiyaro.ai,[35] another model hub for AI apps. The resources are also listed in Meta-Share[36] and tools could also be added to the CLARIN switchboard.[37]

[35] https://console.tiyaro.ai/explore?q=opus-mt &pub=Helsinki-NLP
[36] https://metashare.csc.fi/
[37] https://switchboard.clarin.eu/

3.3.1 PyTorch and the transformers library

The success of deep learning has been made possible thanks to the availability of open software that allows easy adaptation of neural approaches to a wide range of tasks, NLP-related ones being very visible among them. General-purpose frameworks such as PyTorch[38] and Tensorflow[39] provide the essential backbone of most of the work done in this area. Specialized high-level libraries on top of those frameworks nowadays make it easy to get started with state-of-the-art approaches to neural NLP and also enable access to released pre-trained models of various kinds. The `transformers` library published by Hugging Face[40] is one of the most popular hubs of modern NLP, which is largely driven by the community.

With the help of the scientists at Hugging Face, OPUS-MT models have been fully integrated into the `transformers` library by converting them to PyTorch. The impact is significant as this enables a wide range of users to immediately get access to thousands of pre-trained translation models supporting many languages and language pairs. Models are now available from the Hugging Face model hub and can be used with a few lines of code or even through the online inference API (see Fig. 9).

The collaboration with Hugging Face is on-going and future developments will make their way into the popular framework. Recently, conversion tools were adapted to cover more flexibly different architectures. Tensorflow-based backends are also supported now, which creates additional opportunities.

3.3.2 Integration into the European language grid

The European Commission has been one of the most important players in creating resources and solutions for inclusive language technology. The European Language Grid (ELG) is one of the EU-supported initiatives to build a comprehensive infrastructure for NLP resources and tools.[41] OPUS-MT has been funded as one of the ELG pilot projects. This has led to a seamless integration of translation services based on OPUS-MT models in containerized server implementations at ELG.

ELG services can be accessed from their live platform and models are loaded on demand from their cloud infrastructure running through Kubernetes[42] and OpenStack.[43] HTTPS requests can be sent to the internal API and services such as OPUS-MT can also be called programmatically from, for example, a dedicated ELG python library. Metadata records and persistent identifiers based on Digital Object Identifiers (DOI)[44] create sustainable resources according to the Findability, Accessibility,

[38] https://pytorch.org/
[39] https://www.tensorflow.org/
[40] https://huggingface.co/transformers/
[41] https://live.european-language-grid.eu/
[42] https://kubernetes.io/
[43] https://www.openstack.org/
[44] https://www.doi.org/

Interoperability, and Reuse (FAIR) principles.[45] The collaboration with ELG ensures long-term preservation of our developments and provides the necessary maintenance through their standardized infrastructure.

OPUS-MT now includes procedures to generate metadata records and docker images that can be pushed directly to Docker Hub[46] and ELG. Through those routines, new services can easily be registered inside of the ELG platform and become available to end-users and developers after some internal validation period. Docker images are naturally also available outside of the ELG platform and can be fetched and deployed locally or on other cloud services. We also integrate ELG translation services into OPUS-CAT, making it possible to include OPUS-MT in professional translation workflows. More information on OPUS-CAT can be found in Sect. 3.4.

3.3.3 End-user applications

The previous sections already showed several ways of integrating OPUS-MT into end-user applications through online services and containerized server solutions. Further integration into tools described below demonstrate additional use cases and application areas.

Interactive and instant translation is useful for quick access to information in other languages. The Bergamot project[47] created speed-optimised implementations for local translation engines that can run inside of a web browser or in dedicated desktop applications.[48] The main idea is to use knowledge distillation, quantization and lexical shortlists[49] to push the limits of decoding speed. Furthermore, decoder implementations can be optimized in various ways and a customized fork of Marian creates the backbone of the Bergamot solution. OPUS-MT models can be used through the same infrastructure as they are natively built in Marian-NMT. Furthermore, we currently work on systematic distillation of OPUS-MT models to create efficient student models that are compatible with the Bergamot project (see also Sect. 5.2). With those, our models become available in the repository of their browser-based MT solutions[50] and the translateLocally desktop app[51] (see Fig. 10).

Plugins and add-ons for commonly used applications are another means of bringing OPUS-MT to the actual end users. One example is social media channels that are frequently used by millions and even billions of people around the World. As a response to the crisis in Ukraine, we developed a prototype for translating from and to Ukrainian using a translation bot in Telegram. To date, according to

[45] https://www.go-fair.org/fair-principles/

[46] Docker Hub is a repository of user-contributed software container images, see https://hub.docker.com/.

[47] https://browser.mt/

[48] https://translatelocally.com/

[49] A lexical shortlist restricts the output vocabulary to a small subset of translation candidates to improve decoding speed, see https://marian-nmt.github.io/docs/ for further information.

[50] https://translatelocally.com/web/

[51] OPUS-MT fork at https://github.com/Helsinki-NLP/OPUS-MT-app/

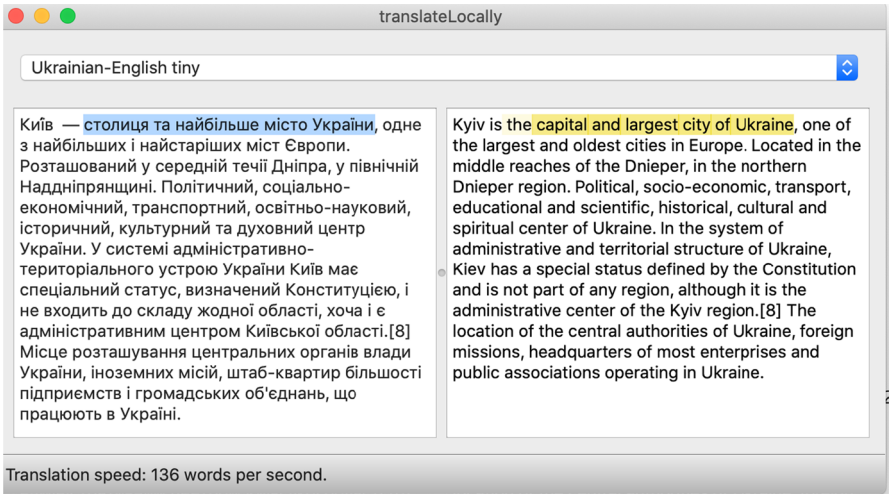

Fig. 10 The translateLocally desktop application developed in the Bergamot project with an adaptation for OPUS-MT models. Here showing the example of Ukrainian–English translation with some cross-lingual highlighting done through the alignment feature

SimilarWeb statistics[52] Telegram has been the most-used messaging platform in Ukraine both in Google Play Store and Apple App Store. The bot is implemented using `aoigram`, a framework for Telegram Bot API written in Python; it is asynchronous, thus multiple requests can be processed almost simultaneously. Interacting with the bot is easy and convenient as sending a message is all that is required to obtain the desired translation. The bot uses our websocket server and supports several source and target languages in connection with Ukrainian. An example of the operation of the bot is shown in Fig. 11.

3.4 Professional workflows with OPUS-CAT

OPUS-CAT[53] is a collection of software packages that enable translators to use pre-trained OPUS-MT models in computer-assisted translation tools. MT is currently routinely used in professional translation work, but the field is dominated by proprietary MT systems offered by large tech companies (such as Google or Microsoft) or specialized machine translation vendors (such as DeepL and ModernMT). OPUS-CAT offers a free and open-source alternative to the proprietary MT products.

OPUS-CAT consists of a local MT engine and a selection of plugins and other types of CAT tool integration. The core of the local MT engine is a Windows build of the Marian framework, which is supplemented by a GUI for downloading and automatically installing OPUS-MT models from a centralized repository. The engine can be simply installed by extracting the installation package, and models

[52] https://similarweb.com/apps/top
[53] https://helsinki-nlp.github.io/OPUS-CAT/

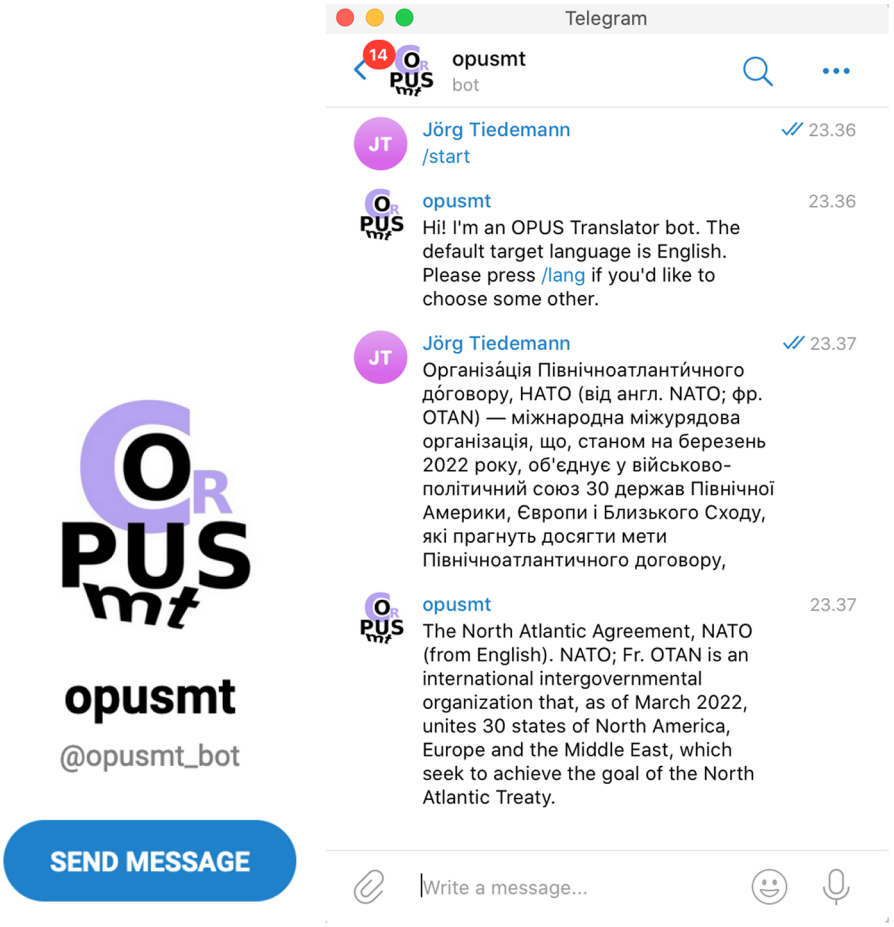

Fig. 11 A simple Telegram translation bot serving the translation from and to Ukrainian. The example shows the translation of a short text snippet taken from Ukrainian Wikipedia

can be searched by language names. The local MT engine does not require a connection to any external service, all the translations are generated on the user's computer using its native CPU. The local MT engine is currently available only for Windows, since many CAT tools are also only available in Windows, and professional translators are therefore likely to be Windows users. Linux or Mac versions of the MT engine are currently under development.

The MT engine GUI provides a simple functionality for translating text, but the translations are mainly intended to be generated via an API that the MT engine exposes. CAT tool plugins and other integrations can be built on top of this API. OPUS-CAT currently supports most of the widely-used CAT tools. It includes

dedicated plugins for three desktop CAT tools: Trados Studio,[54] memoQ,[55] and OmegaT.[56] In some CAT tools, such as Wordfast,[57] OPUS-CAT can be used by connecting directly to its API through a custom MT provider functionality. OPUS-CAT also includes a Chrome browser extension, which makes it possible to use OPUS-MT in browser-based CAT tools. The Chrome extension currently supports Phrase[58] (formerly Memsource) and XTM,[59] and modifying it to support other browser-based CAT tools is relatively simple.

One of the advantages of using OPUS-CAT in professional translation is that it is inherently secure and confidential. Since no external services are used, sensitive data is never at risk. While many commercial MT providers offer on-premises installations similar to OPUS-CAT, such installations are expensive and they cannot be adapted as freely as the open-source OPUS-CAT. The guaranteed confidentiality of OPUS-CAT also makes it possible for individual translators to use it in their work without breaching confidentiality agreements.

OPUS-CAT is intended for professional translators, so it includes functionalities for addressing issues related to MT use in professional translation, such as domain adaptation and tag handling.

The utility of generic NMT models in professional translation is uncertain, while performance improvements resulting from the use of domain-adapted NMT models have been observed multiple times (Läubli et al., 2019; Macken et al., 2020). Because of this, OPUS-CAT MT Engine includes a mode for fine-tuning models with bilingual data. Since fine-tuning has to be performed locally on a CPU, only small amounts of data are used (usually tens of thousands of translations pairs) and the training only lasts for a single epoch. To avoid problems with over-fitting, a very conservative learning rate is used (0.00002, a fifth of the default initial learning rate in Marian).

During fine-tuning, the model is validated against in-domain and out-of-domain validation sets, and the validation results are displayed graphically for the user to allow them to detect potential problems. Despite the conservative training settings and the small amount of data, informal testing and user feedback indicates that the fine-tuning has a noticeable effect on the usability of the MT system.

In professional translation, source documents often contain placeholder tags or tag pairs indicating formatting. Placing tags manually is time-consuming, so MT systems designed for professional translation should ideally place source tags automatically in the generated MT. OPUS-CAT supports two methods of placing tags automatically. When fine-tuning a model, OPUS-CAT can be specified to include tags in the fine-tuning set, enabling the fine-tuned model to learn the correct tag

[54] https://www.trados.com/products/trados-studio/
[55] https://www.memoq.com/
[56] https://omegat.org/
[57] https://www.wordfast.com/
[58] https://phrase.com/
[59] https://xtm.cloud/

placement implicitly. With base models, sub-word alignments (supported by most OPUS-MT models) can also be used to deduct the correct placement of tags.

OPUS-CAT further supports rules that can be used to pre-edit the source text before using it as MT input, or to post-edit the MT output before presenting it to the translator. These rules can be applied to address systematic errors in source texts or in MT output. For instance, a pre-edit rule can be created to change the letter case of the source text or to correct recurring typos, and a post-edit rule can be used to correct a recurring MT mistake.

Deviation from client- or domain-specific terminology is one of the main obstacles to using MT in professional translation. While fine-tuning and the edit rules help to address some of the issues, they have their limitations: fine-tuning does not guarantee the use of correct terminology, it only increases the tendency to use it, and edit rules can be feasibly defined for only a very limited amount of cases. Work is currently in progress to include stronger terminology support in OPUS-CAT by implementing soft terminology constraints based on target lemma annotations (Bergmanis & Pinnis, 2021).

OPUS-CAT is currently the only open-source, free solution for neural machine translation use in professional translation. There is clearly demand for such a solution, as many individual translators currently use OPUS-CAT in their work, and several organizations have included OPUS-CAT in their translation workflows. From the point of view of the wider OPUS-MT project, the significance of OPUS-CAT is that it provides another channel of disseminating the pre-trained models and of gathering feedback and experiences from an important group of MT users, i.e. professional translators.

4 Benchmarks and evaluation

Important in development but also for deployment is quality control and procedures to monitor progress. Benchmarks and evaluation pipelines are essential to fill that need. Fortunately, regular shared tasks in machine translation produce various benchmarks and evaluation strategies, and recently growing interest in low-resourced languages and domains also improves the language coverage of available test sets. Within our ecosystem, we try to systematically collect existing benchmarks and contribute to the collection also with our own efforts, e.g., through the Tatoeba translation challenge mentioned earlier in Sect. 2.4.

One of the crucial questions for the success of OPUS-MT is the quality of the models we release. Regularly monitoring and widely evaluating them is therefore essential. Comparing our models to established benchmarks and test sets is one way of approximating translation quality. OPUS-MT does not try to compete with highly domain-optimized models submitted to specific shared tasks but rather focuses on general-purpose models that can be re-used and refined later. Nevertheless, putting our models in perspective with other results is a good way of demonstrating their applicability. Figure 12 shows the example of Finnish–English translation results measured on the popular news translation task at WMT. The figure shows that our models fare well (in terms of BLEU scores) in comparison to officially submitted

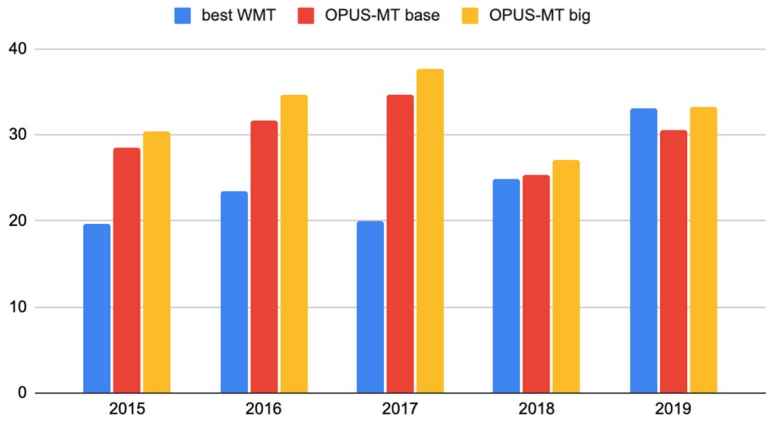

Fig. 12 Comparing OPUS-MT models (in terms of BLEU scores) to official results from the news task at WMT in several years for translating from Finnish to English. WMT scores are taken from http://wmt.ufal.cz/

systems during the evaluation campaign even though they are not directly comparable for various reasons (for example, being trained on different data sets and tuned for different domains).

For practical reasons, we currently focus on automatic evaluation but we also discuss options where community-driven leaderboards can facilitate regular manual evaluations as well. To support systematic benchmarking, we compile test sets into our own repository,[60] which feeds into the OPUS-MT leaderboard described further down in Sect. 4.1. Additionally, we have also developed test suites to complement the general-purpose translation quality assessment addressed by regular MT benchmarks, described in Sect. 4.2.

4.1 The OPUS-MT dashboard

The large number of models we train and the high language coverage and diversity we support makes it necessary to monitor progress and to obtain an overview of the available systems that are among the released packages. A common way to summarize and compare models is to use leaderboards on established benchmarks. OPUS-MT implements a simple interactive dashboard that provides information from our regular benchmarks in terms of tables and bar charts. Figure 13 shows a screenshot from our website[61] with averaged BLEU scores for English–Ukrainian machine translation measured on Flores Goyal et al. (2022) and the Tatoeba benchmark. Currently, we support BLEU (Papineni et al., 2002), spBLEU (Goyal et al., 2022), chrF (Popović, 2015), chrf++ (Popović, 2017) and COMET (Stewart et al., 2020) scores but other measures may be added once they become available from our systematic

[60] https://github.com/Helsinki-NLP/OPUS-MT-testsets/
[61] https://opus.nlpl.eu/dashboard/

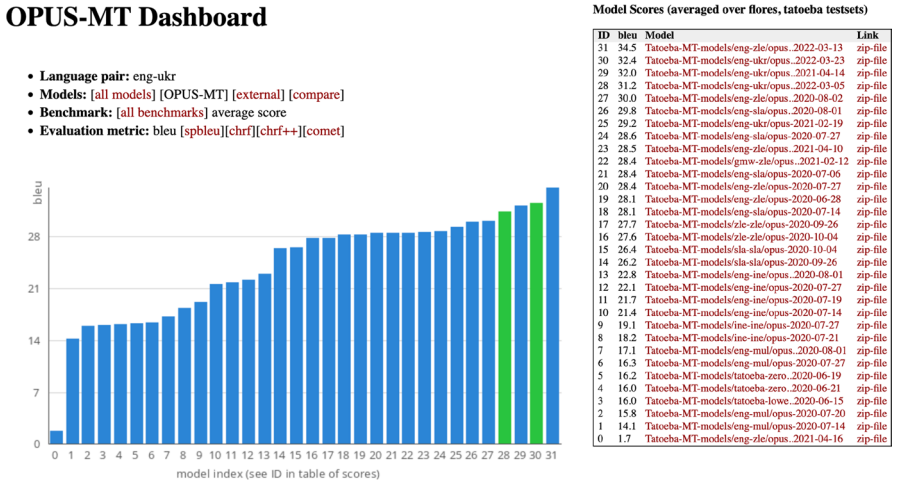

Fig. 13 A screenshot from the OPUS-MT dashboard. The example shows averaged BLEU scores over two benchmarks (Flores and Tatoeba) for the translation from English to Ukrainian. Green bars mark compact student models that can be used for efficient translation

test procedures. The dashboard allows to select language pairs, benchmarks and provides various views on language-specific or model-specific evaluations. The test set translations and the models themselves can be downloaded from the links provided in the table. The texts of the original benchmarks are also linked.

The dashboard takes the information from our score repository (OPUS-MT leaderboard) and currently does not support any dynamic upload of new models or translation results. The ambition is not to provide a fully-fledged system for benchmarking new systems but rather to provide a view on our evaluation results to provide summaries and overviews of OPUS-MT capabilities. Note that we include a large variety of benchmarks in order to provide a comprehensive picture on model performance, which is not restricted to one specific domain and evaluation test set.

Another useful visualization is the interactive geographic map that we generate from our released models.[62] Figure 4 in Sect. 2.4 shows an example of such a plot. As we are striving for a wide language coverage, it is useful to visually see gaps across the globe. We, therefore, plot translation models according to their source or target language onto the geo-location provided by Glottolog.[63] We use the langinfo library[64] to extract the location information from the Glottolog database using the ISO-standard language IDs in our translation models. OpenStreetMap is used to visualize the locations on a map and we indicate the size of the test set by the size of the dot (to illustrate reliability of the score) and use colors on a continuous scale from green (best) to red (worst) to indicate quality in terms of the selected

[62] https://opus.nlpl.eu/NMT-map/
[63] https://glottolog.org/
[64] https://github.com/robertostling/langinfo

benchmark. We base the visualization on chrF scores, which is more reliable across languages than BLEU, but note that even those scores are problematic to compare in general. Only the highest scoring model is shown for each language pair.

The map is interactive and allows to select source or target language to base the illustration on. We also generate maps for various benchmarks and use feature templates to show information about the model that correspond to a dot on the map. From that template, links to the downloadable model releases and further information are available.

4.2 Linguistic test suites

The impressive advances in translation quality seen in recent years have led to a discussion whether translations produced by professional human translators can still be distinguished from the output of NMT systems, and to what extent automatic evaluation measures can reliably account for these differences (Hassan et al., 2018; Läubli et al., 2018; Toral et al., 2018). One answer to this question lies in the development of so-called *test suites* (Burchardt et al., 2017) or *challenge sets* (Isabelle et al., 2017) that focus on particular linguistic phenomena that are known to be difficult to evaluate with simple reference-based metrics such as BLEU. However, most existing test suites require significant amounts of expert knowledge and manual work for compiling the examples, which typically limits their coverage to a small number of translation directions.

To facilitate the development of test suites for a wide range of language pairs, we have introduced MuCoW,[65] a language-independent method for automatically building test suites for lexically ambiguous nouns (Raganato et al., 2019, 2020; Scherrer et al., 2020). MuCoW takes advantage of the parallel corpora available in OPUS and of BabelNet (Navigli and Ponzetto, 2012), a wide-coverage multilingual encyclopedic dictionary obtained automatically from various resources (WordNet and Wikipedia, among others). In a nutshell, the three following steps are needed to create a MuCoW test suite:

1. We identify ambiguous source words and their translations in parallel corpora, matching only those words that are found to be ambiguous in BabelNet.
2. Since lexical resources such as BabelNet are known to suffer from overly fine granularity of their sense inventory, we merge the BabelNet sense clusters with similar meanings, taking advantage of pre-trained sense embeddings: if the cosine similarity between two sense embeddings exceeds a certain threshold, the corresponding clusters are merged (Raganato et al., 2019).
3. To build the test suite properly speaking, we extract sentence pairs from the parallel corpora, making sure that sentences from different corpora are represented. Each sentence is complemented with a list of correct and a list of incorrect translations of the ambiguous source word.

[65] https://github.com/Helsinki-NLP/MuCoW

Table 1 Examples of English–German test suite instances

Example containing **ambiguous word**	Correct translations	Incorrect translations
It occurred to me that my **watch** might be broken	Armbanduhr, Uhr	Wache
I hope you didn't get distracted during your **watch**	Wache	Armbanduhr, Uhr
In winter, the dry leaves fly around in the **air**	Luft, Luftraum, Aura	Miene, Ausdruck
He remained silent for a moment, with a thoughtful but contented **air**	Miene, Ausdruck	Luft, Luftraum, Aura
Harry had to back out of the competition because of a broken **arm**	Arm	Waffe
So does the cop who left his side **arm** in a subway bathroom	Waffe	Arm
Drain the pasta and return the pasta to the **pot**	Blumentopf, Kochtopf, Topf, Nachttopf	Marihuana, Gras
Where did those idiots get all of this **pot** anyhow?	Marihuana, Gras	Blumentopf, Kochtopf, Topf, Nachttopf

The ambiguous English source word is highlighted in bold, and correct and incorrect German translations—as inferred by the MuCoW procedure—are given

Table 1 shows some examples of test suite instances for the English–German translation direction. MuCoW has identified *watch, air, arm* and *pot* as ambiguous English nouns, extracted example sentences using these nouns from various OPUS corpora, and associated each sentence with correct and incorrect German translations.

MuCoW has been first introduced at WMT-19 as a test suite for the News Translation Task, for 9 language pairs (Raganato et al., 2019). In this context, we showed that tuned NMT systems performed well on our evaluation suite, but struggled when dealing with out-of-domain data. We observed the same trend in the following year at WMT-20, with only a general improvement in translation quality for the top-ranked systems (Scherrer et al., 2020).

Finally, we also created a MuCoW benchmark set that includes training data with known sense distributions, to evaluate competing systems on a fair ground (Raganato et al., 2020). Our findings show that state-of-the-art Transformer models struggle when dealing with rare word senses. Interestingly enough, adding more training data, not necessarily containing the ambiguous words of interest, contributes to mitigating such issues. Moreover, we also show that word segmentation does not affect the disambiguation ability much, whereas the performance drops consistently across languages when evaluating sentences from noisy sources.

5 Scaling-up and scaling-down

Increasing language coverage and providing lightweight models that are easy to integrate and deploy are both our priorities for advancing the MT field towards a more open and inclusive state. However, pursuing both objectives in parallel without losing translation performance is still an open problem.

A common approach to multilingual NMT makes use of fully shared models, where a single neural model is trained over parallel data including several translation directions, and all model weights are updated at every training step. The multilingual models obtained with OPUS-MT as described in Sect. 3.1 are examples of this paradigm.

Adding languages to a fully shared multilingual model (Johnson et al., 2017) implies distributing the model capacity over several translation tasks, which may lead to decreased bilingual performance if the number of parameters of the model is kept constant. On the other side, having a large number of bilingual models, or a model architecture with language-specific components, can be impractical without extensive access to HPC facilities. In the first case, the model size can become unmanageable due to the additional language-specific components. In the second case, scaling up language coverage involves handling a large number of (relatively small) NMT model files. In this section, we discuss two strategies that mitigate this issue and give an overview of ongoing experiments.

5.1 Modular NMT

The OPUS ecosystem presents extensive potential for exploring and analyzing the capabilities of multilingual MT systems. Hand in hand with the European Research Council (ERC) funded *Found in Translation* (FoTran) project,[66] we develop natural language understanding models trained on implicit information given by large collections of human translations. Aligning both initiatives, it is in our best interest to distribute and make broadly available, both our toolkit for distributed training of multilingual models and the pre-trained models to be reused and also fine-tuned to new tasks. At training time, highly multilingual models require a large amount of computational power. However, we are building an architecture with a modular design that allows to reuse the trained components (language-specific and shared modules) on relatively small processing units,[67] making multilingual models more affordable and increasing their applicability. In contrast to the OPUS-MT models, this implementation is based on OpenNMT-py[68] (Klein et al., 2020). Our ambition is also to release models from those efforts in the near future but for now, we focus on the development of the modular and scalable framework first.

[66] http://www.helsinki.fi/fotran
[67] https://github.com/Helsinki-NLP/FoTraNMT
[68] https://github.com/OpenNMT/OpenNMT-py

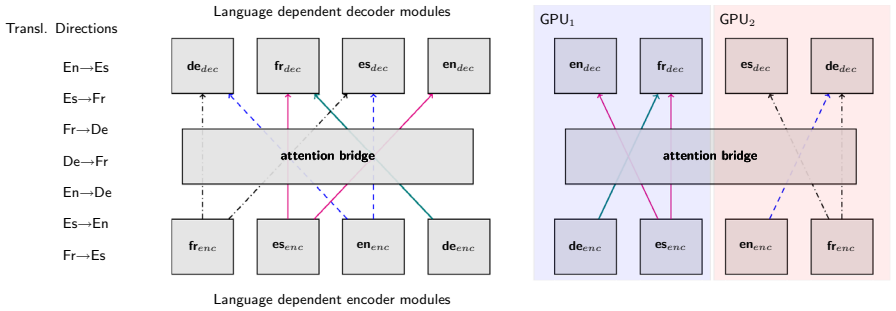

Fig. 14 Diagram of the multilingual attention-bridge model used in a simple example. In this example we use 7 language pairs and 2 GPUs to illustrate the effectiveness of assigning language pairs in different GPUs to reduce inter-device communication

5.1.1 The multilingual attention-bridge model

Our original motivation to build a multilingual translation model in FoTran is to explore the role of "cross-lingual grounding" for resolving ambiguities through translation. The intuition behind this idea is that translations provide a semantic mirror (Dyvik, 2004) reflecting the same meaning with the expressions of another language. We want to explore that signal in representation learning but it also becomes interesting in transfer learning for the original task of machine translation. We designed a modular NN architecture (Vázquez et al., 2019) in which the model incorporates an intermediate shared layer that exploits the semantics from each language while keeping language-specific components.

The architecture is illustrated in the central diagram of Fig. 14. It follows a traditional sequence-to-sequence encoder–decoder architecture, but incorporates language specific encoders and decoders that are connected through a shared component to enable multilingual training and knowledge transfer. We obtain multilingual representations generated by the encoders by forcing the information to flow through a bottleneck inner-attention layer connecting all the language-specific modules (Vázquez et al., 2020). This layer summarizes the encoder information in a fixed-size (language-agnostic) meaning representation, which is useful for machine translation (including the support of zero-shot scenarios) and downstream tasks that require semantic reasoning and inference. Experimental results point towards the improvement of both the translation quality, and the abstractions acquired by our model when including more languages (Vázquez et al., 2019; Raganato et al., 2019).

5.1.2 Scaling up for high linguistic diversity

In its basic implementation, our multilingual model requires high computational resources at training time due to the use of language-specific modules. Scaling up the number of translation directions on a single device is restricted by the memory limits of that specific computing node.

To address those challenges, we implemented a multi-node and multi-GPU training setup that incorporates the following strategies: (1) distribute efficiently the modules across several processing units, (2) train the network over many translation directions reducing memory overhead, and (3) reuse the trained modules without having to load the entire network. Taken together, these strategies deliver a cost-effective multilingual NMT system that can further be used for extracting multilingual meaning representations.

We distribute the model across multiple processing units by loading, in each device, encoders and decoders for only a subset of the translation directions. The inner-attention layer is shared across all processing units. All modules that are present in more than one device are initialized with the same weights. Parameters that are present in more than one device need to remain synchronous at each training step. We schedule the gradient communication of all parameters to reduce the inter communication load.

In general, allocating language pairs with common source/target languages on the same device decreases both the total memory footprint of the model and the amount of communication needed to keep the modules synced. To see this, we can follow the example in Fig. 14. In the example, there are 7 language pairs to be trained simultaneously, and we have access to 2 GPUs. Defining a partition like the one on the right-most minimizes the amount of communication between the devices to keep the modules synced, reducing the training time.

Gathering together language pairs based on the source (or target) language could result in a scattered configuration depending on the target (or source) languages included. When dealing with a high number of translation directions (and a limited number of source and target languages) it becomes impossible to avoid this condition. We address these problems using two strategies. First, we solve the allocation problem to minimize inter-device communication. Since in most cases the problem has no feasible exact solution, we approximate it using a local search algorithm over a cost matrix that makes it cheaper to assign the same language to a given GPU. Second, we propose to schedule the gradient updates to minimize the waiting time when inter-device communication happens.

With this infrastructure at hand, we currently work on scaling up experiments to highly multilingual models that we can train on large high-performance clusters with a wide distribution of components over compute nodes and massively parallel training data. We aim at a large language coverage and will also release the model with separate components that can individually be loaded for efficient inference and further fine-tuning or downstream testing.

5.2 Knowledge distillation

Machine translation requires optimization for speed and size to be available in practical solutions and on various devices. OPUS-MT models are already compact and fast in comparison to the ever-growing multilingual language models that are often referred to in recent NLP work. Nevertheless, there are various ways to further improve decoding speed while reducing resource requirements. Knowledge

distillation (Hinton et al., 2015) is one of the popular techniques that reduces complex neural architectures (used as a so-called teacher model) to compact student models, which obtain the essential information from the generalization the teacher model has learned previously in a typically very expensive and time-consuming learning process.

We use sequence-level knowledge distillation (Kim & Rush, 2016), in which a teacher model provides complete translations of some training data, and the student model learns the task on that output rather than the original reference translations. For simplicity, we "forward-translate" the training data with the teacher model using a narrow-width beam search and only use the best translation for teaching the student. Currently, we do not apply ensemble methods either, which would push the output quality of the teacher a bit further. However, we apply normalized cross-entropy filtering (Behnke et al., 2021) to remove some translation noise using a reverse translation model to score the translations obtained by the teacher model. This score indicates the level of "hallucination", i.e. how much the translation diverges from the original input in a sense that it cannot reliably be reconstructed from the translation produced by the system. Following related work, we retain 95% of the data that have been sorted by that score.

In our current experiments, we look at different network architectures to study the impact of distillation. In particular, we use a base transformer model with 6 layers in the decoder and then reduce the decoder from a 6-layer transformer model to an RNN-based variant using simpler simple recurrent units (SSRU) (Kim et al., 2019) with two stacked layers (henceforth called "small"). Following previous work (Behnke et al., 2021), we also test the reduction of the embedding size to 256 dimensions (from 512 in default settings) and try two different variants ("tiny" and "tiny11") that differ in the size of the transformer feed-forward network parameter (1,536 in "tiny11" instead of the default 2048 in "tiny") and the size of the encoder (3 layers in "tiny" instead of 6).

A drop in performance can be expected when reducing the network and the parameters size of the model. However, student models are known to recover well from the reduced capacity and, therefore, create efficient alternatives to more expensive models. Figure 15 shows our results in terms of BLEU scores for Finnish–English MT for the various models we tried in comparison to standard non-distilled ones. We can see that in all cases the regular model of corresponding size is outperformed by the student distilled from a larger teacher model and, especially important, smaller student models substantially recover from the drop in performance we see with reduced model capacities in regular training from scratch. In particular, small students reach more or less the same performance as the much bigger teacher model, which is a remarkable achievement.

The effect of model reduction can be seen in terms of size and decoding speed. Table 2 summarizes properties and benchmarks on the Tatoeba test set with 10,000 sentences and over 48,000 words for the Finnish–English models discussed above. The space requirements go down dramatically and the smallest model is just about 23 MB in size, about 10% of the big transformer model we used as a teacher (12 layers each in encoder and decoder). We also see the importance of quantization (using 8-bit integers, `int8`, in this case). Another

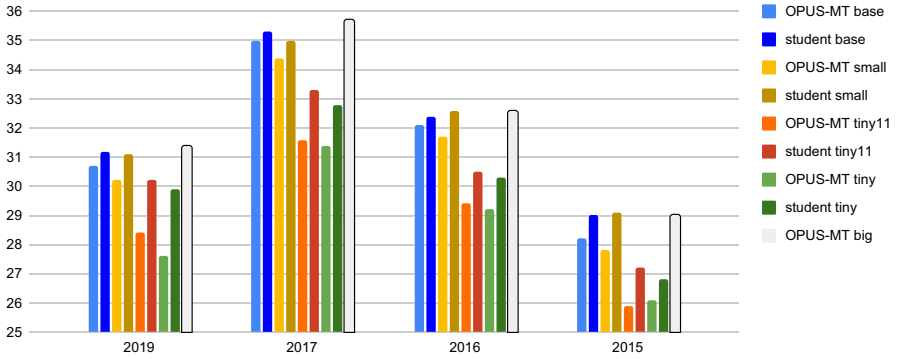

Fig. 15 Comparing regular NMT models (labeled as OPUS-MT) to distilled student models created through sequence-level knowledge distillation for Finnish–English MT models of various sizes. All distilled models use the same teacher model (transformer-big, last bar in each group) and regular models are trained from scratch with the same architectures as the corresponding student models. The years in the X-axis refer to different news test sets from WMT and the scores refer to BLEU scores

Table 2 Comparison of different models for translating from Finnish to English

Model	Compressed model size		Decoding speed	
	Original	Quantized	4 CPUs	+Shortlist
Big	891 MB	224 MB		
Base	294 MB	74 MB	46.36s	40.47s
Small	226 MB	57 MB	24.07s	17.89s
Tiny11	96 MB	25 MB	10.98s	7.24s
Tiny	89 MB	23 MB	9.90s	6.22s

Decoding speed is measured on the Tatoeba test set with 10,000 sentences covering approximately 48,000 words. Model names refer to the same settings as in Fig. 15. Model size refers to gzipped file size (with and without model quantization). Decoding is done on a CPU node with 4 cores with and without lexical shortlists (see Sect. 3.3.3)

substantial improvement can be seen in decoding speed: The smallest model is able to decode the entire test set in less than 10 s on a 4-core CPU machine, an increase in speed by more than factor 4 compared to the base transformer model. Additional lexical shortlists, compiled from 100 top aligned tokens on the training data, make it possible to push the time down to about 6 s.

With these levels of performance and size, we can afford real-time translation on regular and even small devices. The browser-based MT solutions and local desktop apps discussed in Sect. 3.3 become feasible solutions with such highly-optimized distilled student models and the models are compatible with the Bergamot-derived systems and can immediately applied in applications based on that project (see, e.g. https://translatelocally.com/web/).

6 Related work

OPUS and OPUS-MT are certainly not alone on the quest for open and transparent machine translation. Numerous projects and initiatives have been created over the years and listing them all is beyond the scope of this section. One of the issues with project-related work is the long-term perspective and the creation of sustainable platforms with a clear continuation.

During the era of statistical machine translation, many projects have been related to the framework of Moses,[69] leading to a rich infrastructure of resources and tools. Projects to be mentioned include EuroMatrix, EuroMatrixPlus,[70] TC-STAR,[71] LetsMT![72] and MateCAT.[73]

Another long-lasting infrastructure from the paradigm of rule-based machine translation is connected to Apertium.[74] Compared to most developments driven by EU projects that typically result in commercial exploitation, the development of Apertium always emphasizes its place as an open-source and free translation platform. In that sense, it is the most related work to OPUS-MT, but with a focus on a different translation approach and a much longer history.

OPUS and OPUS-MT build on top of many other open-source and data-sharing efforts. EU-funded research projects and resource development supported the creation of resources and tools in projects like ParaCrawl,[75] GoURMET,[76] MT4ALL[77] and MaCoCu.[78]

Open software platforms are also essential for our project and the rapid development of neural MT could not have been possible without software packages such as OpenNMT[79] and Marian-NMT,[80] to name just two of them with relevance to our project. Translation efficiency was a focus in the Bergamot project,[81] and the work continues in the HPLT project,[82] which has close ties to OPUS and OPUS-MT.

Language coverage in NLP and inclusivity are also addressed by various regional and international initiatives. Masakhane,[83] for example, is a grass-roots organization for African NLP, which includes dedicated work on resource and tool development for translating African languages. From the Nordic perspective, one can mention

[69] http://www2.statmt.org/moses/
[70] https://www.euromatrixplus.net/
[71] http://www.tcstar.org/
[72] http://project.letsmt.eu/
[73] https://www.matecat.com/
[74] https://www.apertium.org/
[75] https://www.paracrawl.eu/
[76] https://gourmet-project.eu/
[77] http://ixa2.si.ehu.eus/mt4all/project.html
[78] https://macocu.eu/
[79] https://opennmt.net/
[80] https://marian-nmt.github.io/
[81] https://browser.mt/
[82] https://hplt-project.org/
[83] https://www.masakhane.io/

Giellatekno,[84] with their focus on Sami language technology. The EU also tries to push for more comprehensive data sharing and resource building with the European Language Resource Coordination (ELRC)[85] and their digital strategy on creating accessible data spaces. Resource repositories like Meta-Share[86] have been developed and filled with information, links and metadata, and resource infrastructures such as CLARIN[87] play an important role in the coordination of such efforts.

Another push for open machine translation certainly also comes from commercial research labs and NLP startups. Research at big tech companies supports the development of platforms such as fairseq,[88] tensor2tensor[89] and Sockeye.[90] Important multilingual resources such as WikiMatrix and CCMatrix have their origin in those labs and made their way into OPUS. The No Language Left Behind project[91] together with extended benchmarks[92] further pushed the frontiers of multilingual machine translation. Another central point for NLP research nowadays is Hugging Face together with their popular `transformers` library and the growing collection of data sets[93] and models.[94] The availability of all resources in one hub makes it easy and straightforward to get started with NLP research.

Another recent initiative to be mentioned in connection with machine translation is LibreTranslate.[95] Their efforts are very much in line with OPUS and OPUS-MT, and we will explore collaboration possibilities in future work to join their and other great open-source initiatives that appear in modern NLP.

7 Conclusions

This paper provides an overview of OPUS-MT and its embedding into the OPUS ecosystem. We describe various components that facilitate data curation, model development and machine translation integration into various platforms and applications. Our initiative emphasizes a large language coverage and focuses on public resources and open-source solutions in order to create a transparent and widely applicable support for machine translation. The efforts of OPUS-MT already produced a large number of high-quality pre-trained NMT models that are ready to be used and adapted to various needs in research and practical application development. With this, the project supports a sustainable infrastructure that enables reuse

[84] https://giellatekno.uit.no
[85] https://lr-coordination.eu/
[86] http://www.meta-share.org/
[87] https://www.clarin.eu/
[88] https://github.com/facebookresearch/fairseq
[89] https://github.com/tensorflow/tensor2tensor
[90] https://github.com/awslabs/sockeye
[91] https://ai.facebook.com/research/no-language-left-behind/
[92] https://github.com/facebookresearch/flores
[93] https://huggingface.co/datasets
[94] https://huggingface.co/models
[95] https://libretranslate.com/

of computationally expensive components. Giving access to the entire output of our project enables us to make efficient machine translation available for a wide range of users and NLP developers, without the need of high-performance IT infrastructure to train complex neural models from scratch. Democratizing MT in this way is a major step in the direction of an inclusive information society where language barriers do not lead to significant disadvantages. OPUS-MT contributes to this mission with its community-driven open source initiative.

Our future plans include the development of practical solutions for modular NMT in a highly multilingual setup and further advances in transfer learning and model efficiency. We also continue our efforts in data collection and curation and aim to further increase language support and coverage. Furthermore, we also want to include paragraph- or document-level translation models in OPUS-MT and integrate other advances that can be pushed into production-ready solutions.

Acknowledgements The research presented in this paper was supported by the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement no. 771113), the European Language Grid project through its open call for pilot projects with funding from the European Union's Horizon 2020 Research and Innovation program under grant agreement no. 825627 (ELG) and the Swedish Culture Foundation (Svenska Kulturfonden) in Finland under the grant agreement no. 139592. Continued support is provided by the EU Horizon project HPLT (grant agreement no. 101070350) and GreenNLP funded by the Academy of Finland (project ID 353164). The work was also supported by the NVIDIA AI Technology Center (NVAITC) and we would like to thank NVIDIA for their hardware grants providing GPU cards for research and development. Finally, we like to acknowledge the great support by CSC, the Finnish IT Center for Science Ltd., providing extensive computing and storage facilities used in this research and all projects associated with it.

Funding Open Access funding provided by University of Helsinki (including Helsinki University Central Hospital).

References

Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., Chen, M.X., Cao, Y., Foster, G., Cherry, C., Macherey, W., Chen, Z., & Wu, Y. (2019). Massively multilingual neural machine translation in the wild: Findings and challenges. https://arxiv.org/abs/1907.05019

Artetxe, M., & Schwenk, H. (2019). Margin-based parallel corpus mining with multilingual sentence embeddings. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics* (pp. 3197–3203). Association for Computational Linguistics. https://aclanthology.org/P19-1309

Aulamo, M., Sulubacak, U., Virpioja, S., & Tiedemann, J. (2020). OpusTools and parallel corpus diagnostics. In *Proceedings of the 12th language resources and evaluation conference* (pp. 3782–3789). European Language Resources Association. https://aclanthology.org/2020.lrec-1.467

Aulamo, M., Virpioja, S., & Tiedemann, J. (2020). OpusFilter: A configurable parallel corpus filtering toolbox. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics: System demonstrations* (pp. 150–156). Association for Computational Linguistics. https://aclanthology.org/2020.acl-demos.20

Behnke, M., Bogoychev, N., Aji, A.F., Heafield, K., Nail, G., Zhu, Q., Tchistiakova, S., van der Linde, J., Chen, P., Kashyap, S., & Grundkiewicz, R. (2021). Efficient machine translation with model pruning and quantization. In *Proceedings of the sixth conference on machine translation* (pp. 775–780). Association for Computational Linguistics. https://aclanthology.org/2021.wmt-1.74

Bergmanis, T., & Pinnis, M. (2021). Facilitating terminology translation with target lemma annotations. In *Proceedings of the 16th conference of the European Chapter of the Association for Computational Linguistics* (Main Volume, pp. 3105–3111). Association for Computational Linguistics. https://aclanthology.org/2021.eacl-main.271

Burchardt, A., Macketanz, V., Dehdari, J., Heigold, G., Peter, J.-T., & Williams, P. (2017). A linguistic evaluation of rule-based, phrase-based, and neural MT engines. *The Prague Bulletin of Mathematical Linguistics, 108*, 159–170.

Dou, Z.-Y., & Neubig, G. (2021). Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th conference of the European Chapter of the Association for Computational Linguistics* (Main Volume, pp. 2112–2128). Association for Computational Linguistics. https://aclanthology.org/2021.eacl-main.181

Dyvik, H. (2004). Translations as semantic mirrors: From parallel corpus to wordnet. In *Advances in corpus linguistics, papers from the 23rd international conference on english language research on computerized corpora* (*ICAME*) (Vol. 49, pp. 309–326). Brill. https://doi.org/10.1163/9789004333710_019

Goyal, N., Gao, C., Chaudhary, V., Chen, P.-J., Wenzek, G., Ju, D., Krishnan, S., Ranzato, M., Guzmán, F., & Fan, A. (2022). The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics, 10*, 522–538. https://doi.org/10.1162/tacl_a_00474

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., Liu, S., Liu, T.-Y., Luo, R., Menezes, A., Qin, T., Seide, F., Tan, X., Tian, F., Wu, L., Wu, S., Xia, Y., Zhang, D., Zhang, Z., & Zhou, M. (2018). Achieving human parity on automatic Chinese to English news translation. https://arxiv.org/abs/1803.05567

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531

Hoang, V.C.D., Koehn, P., Haffari, G., & Cohn, T. (2018). Iterative back-translation for neural machine translation. In *Proceedings of the 2nd workshop on neural machine translation and generation* (pp. 18–24). Association for Computational Linguistics. https://aclanthology.org/W18-2703

Isabelle, P., Cherry, C., & Foster, G. (2017). A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2486–2496). Association for Computational Linguistics. https://aclanthology.org/D17-1263

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., & Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics, 5*, 339–351. https://doi.org/10.1162/tacl_a_00065

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). FastText.zip: Compressing text classification models. arXiv:1612.03651

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th conference of the European Chapter of the Association for Computational Linguistics* (Volume 2: Short Papers, pp. 427–431). Association for Computational Linguistics. https://aclanthology.org/E17-2068

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A.F.T., & Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations* (pp. 116–121). Association for Computational Linguistics. http://www.aclweb.org/anthology/P18-4020

Kim, Y.J., Junczys-Dowmunt, M., Hassan, H., Fikri Aji, A., Heafield, K., Grundkiewicz, R., & Bogoychev, N. (2019). From research to production and back: Ludicrously fast neural machine translation.

In *Proceedings of the 3rd workshop on neural generation and translation* (pp. 280–288). Association for Computational Linguistics. https://aclanthology.org/D19-5632

Kim, Y., & Rush, A.M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 1317–1327). Association for Computational Linguistics. https://aclanthology.org/D16-1139

Klein, G., Hernandez, F., Nguyen, V., & Senellart, J. (2020). The OpenNMT neural machine translation toolkit: 2020 edition. In *Proceedings of the 14th conference of the Association for Machine Translation in the Americas* (Vol 1: Research Track, pp. 102–109). Association for Machine Translation in the Americas. https://aclanthology.org/2020.amta-research.9

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., & Herbst, E. (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* (pp. 177–180). Association for Computational Linguistics. https://aclanthology.org/P07-2045

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th annual meeting of the Association for Computational Linguistics* (Vol 1: Long Papers, pp. 66–75). Association for Computational Linguistics. https://aclanthology.org/P18-1007

Kudo, T., & Richardson, J. (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* (pp. 66–71). Association for Computational Linguistics. https://aclanthology.org/D18-2012

Läubli, S., Amrhein, C., Düggelin, P., Gonzalez, B., Zwahlen, A., & Volk, M. (2019). Post-editing productivity with neural machine translation: An empirical assessment of speed and quality in the banking and finance domain. In *Proceedings of machine translation summit XVII: Research track* (pp. 267–272). European Association for Machine Translation. https://aclanthology.org/W19-6626

Läubli, S., Sennrich, R., & Volk, M. (2018). Has machine translation achieved human parity? A case for document-level evaluation. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 4791–4796). Association for Computational Linguistics. https://aclanthology.org/D18-1512

Lui, M., & Baldwin, T. (2012). langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations* (pp. 25–30). Association for Computational Linguistics. https://aclanthology.org/P12-3005

Macken, L., Prou, D., & Tezcan, A. (2020). Quantifying the effect of machine translation in a high-quality human translation production process. *Informatics*. https://doi.org/10.3390/informatics7020012

Marie, B., Fujita, A., & Rubino, R. (2021). Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing* (Vol 1: Long Papers, pp. 7297–7306). Association for Computational Linguistics. https://aclanthology.org/2021.acl-long.566

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence, 193*, 217–250.

Östling, R., & Tiedemann, J. (2016). Efficient word alignment with Markov Chain Monte Carlo. *The Prague Bulletin of Mathematical Linguistics, 106*, 125–146.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://aclanthology.org/P02-1040

Popović, M. (2015). chrF: Character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation* (pp. 392–395). Association for Computational Linguistics. https://aclanthology.org/W15-3049

Popović, M. (2017). chrF++: Words helping character n-grams. In *Proceedings of the second conference on machine translation* (pp. 612–618). Association for Computational Linguistics. https://aclanthology.org/W17-4770

Raganato, A., Scherrer, Y., & Tiedemann, J. (2019). The MuCoW test suite at WMT 2019: Automatically harvested multilingual contrastive word sense disambiguation test sets for machine translation. In *Proceedings of the fourth conference on machine translation* (Vol 2: Shared Task Papers, Day

1, pp. 470–480). Association for Computational Linguistics. https://www.aclweb.org/anthology/W19-5354

Raganato, A., Scherrer, Y., & Tiedemann, J. (2020). An evaluation benchmark for testing the word sense disambiguation capabilities of machine translation systems. In *Proceedings of The 12th language resources and evaluation conference* (pp. 3668–3675). European Language Resources Association. https://www.aclweb.org/anthology/2020.lrec-1.452

Raganato, A., Vázquez, R., Creutz, M., & Tiedemann, J. (2019). An evaluation of language-agnostic inner-attention-based representations in machine translation. In *Proceedings of the 4th workshop on representation learning for NLP* (*RepL4NLP-2019*) (pp. 27–32). Association for Computational Linguistics. https://aclanthology.org/W19-4304

Sánchez-Cartagena, V.M., Bañón, M., Ortiz-Rojas, S., & Ramírez, G. (2018). Prompsit's submission to WMT 2018 parallel corpus filtering shared task. In *Proceedings of the third conference on machine translation: Shared task papers* (pp. 955–962). Association for Computational Linguistics. https://aclanthology.org/W18-6488

Scherrer, Y., Raganato, A., & Tiedemann, J. (2020). The MUCOW word sense disambiguation test suite at WMT 2020. In *Proceedings of the Fifth Conference on Machine Translation* (pp. 365–370). Association for Computational Linguistics. https://aclanthology.org/2020.wmt-1.40

Sennrich, R., Haddow, B., & Birch, A. (2016a) Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics* (Vol 1: Long Papers, pp. 1715–1725). Association for Computational Linguistics. https://aclanthology.org/P16-1162

Sennrich, R., Haddow, B., & Birch, A. (2016b). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics* (Vol 1: Long Papers, pp. 86–96). Association for Computational Linguistics. https://aclanthology.org/P16-1009

Siivola, V., Hirsimäki, T., & Virpioja, S. (2007). On growing and pruning Kneser-Ney smoothed n-gram models. *IEEE Transactions on Audio, Speech and Language Processing, 15*(5), 1617–1624.

Stewart, C., Rei, R., Farinha, C., & Lavie, A. (2020). COMET—Deploying a new state-of-the-art MT evaluation metric in production. In *Proceedings of the 14th conference of the association for machine translation in the Americas* (Vol 2: User Track, pp. 78–109). Association for Machine Translation in the Americas. https://aclanthology.org/2020.amta-user.4

Tiedemann, J. (2009). News from OPUS–A collection of multilingual parallel corpora with tools and interfaces. *Recent Advances in Natural Language Processing, V*, 237–248.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proceedings of the eighth international conference on language resources and evaluation* (*LREC'12*) (pp. 2214–2218). European Language Resources Association (ELRA), Istanbul, Turkey . http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf

Tiedemann, J. (2020). The Tatoeba translation challenge—Realistic data sets for low resource and multilingual MT. In *Proceedings of the fifth conference on machine translation* (pp. 1174–1182). Association for Computational Linguistics. https://aclanthology.org/2020.wmt-1.139

Tiedemann, J., & Nygaard, L. (2004). The OPUS corpus—Parallel and free: http://logos.uio.no/opus. In *Proceedings of the fourth international conference on language resources and evaluation* (*LREC'04*) (pp. 1183–1186). European Language Resources Association (ELRA), Lisbon, Portugal . http://www.lrec-conf.org/proceedings/lrec2004/pdf/320.pdf

Tiedemann, J., & Thottingal, S. (2020). OPUS-MT—Building open translation services for the world. In *Proceedings of the 22nd annual conference of the European Association for Machine Translation* (pp. 479–480). European Association for Machine Translation, Lisboa, Portugal. https://aclanthology.org/2020.eamt-1.61

Toral, A., Castilho, S., Hu, K., & Way, A. (2018). Attaining the unattainable? Reassessing claims of human parity in neural machine translation. In *Proceedings of the third conference on machine translation: Research papers* (pp. 113–123). Association for Computational Linguistics. https://aclanthology.org/W18-6312

Vázquez, R., Raganato, A., Creutz, M., & Tiedemann, J. (2020). A systematic study of inner-attention-based sentence representations in multilingual neural machine translation. *Computational Linguistics, 46*(2), 387–424. https://doi.org/10.1162/coli_a_00377

Vázquez, R., Raganato, A., Tiedemann, J., & Creutz, M. (2019). Multilingual NMT with a language-independent attention bridge. In *Proceedings of the 4th workshop on representation learning for*

NLP (*RepL4NLP-2019*) (pp. 33–39). Association for Computational Linguistics. https://aclanthology.org/W19-4305

Vázquez, R., Sulubacak, U., & Tiedemann, J. (2019). The University of Helsinki submission to the WMT19 parallel corpus filtering task. In *Proceedings of the fourth conference on machine translation* (Volume 3: Shared Task Papers, Day 2, pp. 294–300). Association for Computational Linguistics. https://aclanthology.org/W19-5441

Virpioja, S., Smit, P., Grönroos, S.-A., & Kurimo, M. (2013). Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Report 25/2013 in Aalto University publication series science + technology, Department of Signal Processing and Acoustics, Aalto University, Helsinki, Finland

Xu, H., & Koehn, P. (2017). Zipporah: A fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2945–2950). Association for Computational Linguistics. https://www.aclweb.org/anthology/D17-1319

Zipf, G. K. (1932). *Selected studies of the principle of relative frequency in language*. Harvard University Press.

Authors and Affiliations

Jörg Tiedemann[1] · Mikko Aulamo[1] · Daria Bakshandaeva[1] · Michele Boggia[1] · Stig-Arne Grönroos[1,2] · Tommi Nieminen[1] · Alessandro Raganato[3] · Yves Scherrer[1,4] · Raúl Vázquez[1] · Sami Virpioja[1]

✉ Jörg Tiedemann
jorg.tiedemann@helsinki.fi

Mikko Aulamo
mikko.aulamo@helsinki.fi

Daria Bakshandaeva
daria.bakshandaeva@helsinki.fi

Michele Boggia
michele.boggia@helsinki.fi

Stig-Arne Grönroos
stig-arne.gronroos@helsinki.fi

Tommi Nieminen
tommi.nieminen@helsinki.fi

Alessandro Raganato
alessandro.raganato@unimib.it

Yves Scherrer
yves.scherrer@helsinki.fi

Raúl Vázquez
raul.vazquez@gmail.com

Sami Virpioja
sami.virpioja@helsinki.fi

[1] Department of Digital Humanities, University of Helsinki, Unioninkatu 40, 00014 Helsinki, Finland

[2] Silo.AI, Fredrikinkatu 57 C, 00100 Helsinki, Finland

[3] Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca 336, 20126 Milan, Italy

[4] Language Technology Group, Department of Informatics, University of Oslo, Gaustadalléen 23B, 0373 Oslo, Norway