OPUS-CAT: A State-of-the-Art Neural Machine Translation Engine on Your Local Computer

Neural machine translation (NMT) is one of the success stories of deep learning and artificial intelligence. Revolutionary innovations in the computational architectures made in 2015–2017 have led to dramatic improvements in the quality of machine translation (MT) and changed the field forever. Some professional translators welcome these changes with enthusiasm, others less so. But everyone has to deal with them. Historically, the relationship between human translation and MT has been uneasy and complicated, but an increasing number of players in both fields are now coming to view it as synergistic.[1]

To leverage this synergy, MT developers, disappointed with the quality of crowdsourced evaluation of MT output, are heeding advice from professional translators. The latter, in turn, are using MT suggestions alongside translation memory (TM) matches to facilitate work on their projects, especially since many computer-assisted translation (CAT) tools now have plugins for popular generic MT engines. And one should never forget that the multilingual data used to train MT systems has, at the end of the day, a very human origin.

Most general-purpose MT systems are trained on everything they can find, scrape, clean, and align from the huge amount of multilingual data available. But most of professional human translation is specialized, which is what gives it a real value. Yet the desire to automate everything is unstoppable. Tech giants, custom MT providers, and larger language services providers are actively working, even as you read this, on domain adaptation—the fine-tuning of MT systems to particular areas such as software localization, clinical trials, or pharmaceuticals. Domain adaptation is a hot topic in academic MT research and figures prominently at the annual machine translation competitions.

To make use of the MT adaptation offered by commercial companies, translators are encouraged to upload their TMs and term bases to a cloud-based engine in the hope that the system will learn, for example, to translate "conductor" as "electric conductor" and not "music conductor." Normally this requires a paid subscription, with confidentiality of the uploaded resources promised in return. There's no reason to distrust such assurances.[2] Yet some translators may hesitate to press the *Upload* button and commit their golden super-confidential resources, resulting from years of painstaking work and generating steady revenue, to the cloud. And some may want to know more about what happens, technically speaking, to the uploaded data on the other side of the internet connection.

One could handle all such issues by installing, training, and using a customizable NMT system, such as OpenNMT or Marian NMT (see the links in the sidebar on page 37), on a local machine. But one should be prepared to use command line programming and deal with very intimidating installation and debugging pipelines. Even if the process is successful, before the system is fine-tuned for a specific domain (e.g., oil and gas, aerospace engineering) it needs to be trained from scratch on 10+ million parallel sentences from generic bilingual corpora, such as OPUS. Doing it on a central processing unit (CPU) machine would take several months (and you would need to put it in a freezer). Using a graphics processing unit might be a better option. But to harness its full power, you need to know what to do with it—another pain. It's safe to say that 99% of us don't have the requisite skills to do any of that.

What if someone else did all of that for us? And we could simply enjoy having a pre-trained state-of-the-art NMT system on our PC, totally free, in a familiar Windows environment giving us an opportunity to fine-tune it with our local and exclusive resources? And even better—integrate it into our CAT tools? Sounds too good to be true? Enter OPUS-CAT.

What's OPUS-CAT?

OPUS is one of the largest collections of publicly available bilingual corpora in many language pairs widely used for training MT systems. OPUS-MT, an ongoing project led by Jörg Tiedemann, a professor of language technology at the University of Helsinki, and funded by several European Union and local agencies, is a growing repository of over a thousand pre-trained NMT models intended to be used with Marian NMT, an efficient and robust framework written in C++, which can run on Windows computers. OPUS-CAT is a collection of MT tools built around Marian NMT and developed by Tommi Nieminen[3], an experienced professional translator and MT researcher. It's largely due to his unique combination of skills that we have OPUS-CAT at our fingertips.

Figure 1: OPUS-CAT architecture: From: Nieminen 2021

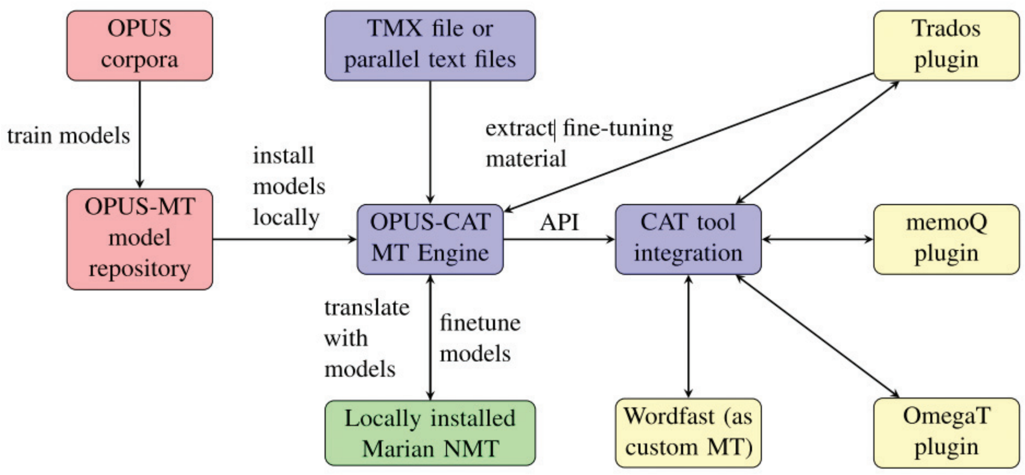
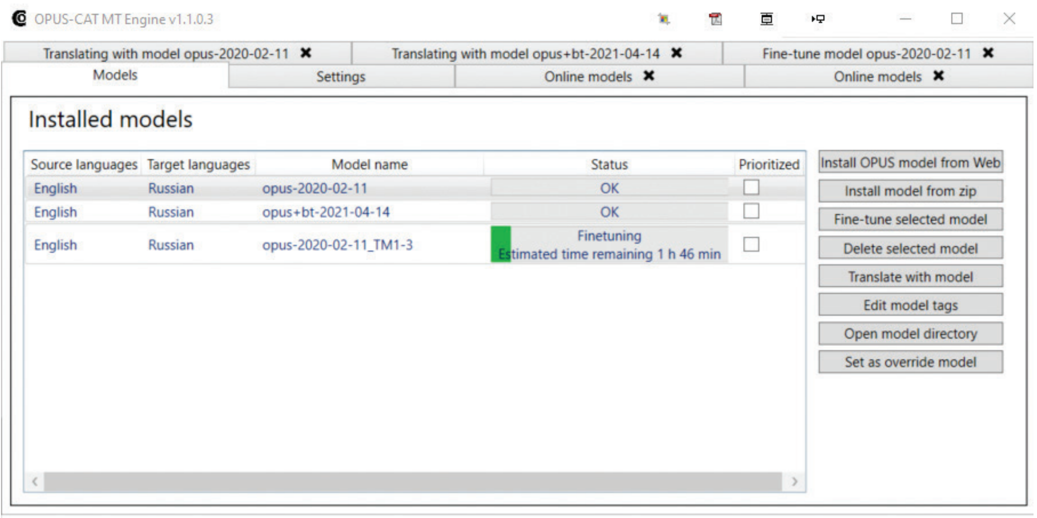


Figure 2: OPUS-CAT graphical user interface



Source languages	Target languages	Model name	Status	Prioritized
English	Russian	opus-2020-02-11	OK	☐
English	Russian	opus+bt-2021-04-14	OK	☐
English	Russian	opus-2020-02-11_TM1-3	Finetuning Estimated time remaining 1 h 46 min	☐

OPUS-CAT Architecture

OPUS-CAT must be installed first as a standalone Windows application. (Please follow the detailed instructions on the OPUS-CAT GitHub site referenced in the sidebar on page 37.) It can then be used as such, or in combination with a CAT tool via a plugin that needs to be installed and configured separately. (See Figure 1.) Integration with CAT tools is an explicit goal of OPUS-CAT that's even reflected in its name, and probably the best way to use it in actual translation projects with optimal speed. Because of Trados Studio's more advanced plugin support, its users may currently enjoy the most benefits from such an integration. But plugins are also available for other CAT tools such as memoQ and OmegaT, along with more limited application programming interface support for Wordfast. The features and documentation of these plugins are in development, as are additional plugins.

The following will focus on the backbone of OPUS-CAT—a locally installed MT engine that enables the deployment and fine-tuning of NMT models in the form of Windows executables built on the Marian-NMT framework.

OPUS-CAT MT Engine

The engine has a minimalistic graphical user interface (GUI) and comes in a zipped folder whose contents can be placed anywhere on your computer. Figure 2 shows the GUI for the most recent version of OPUS-CAT available at the time I'm writing this article. The GUI may look slightly different in later versions.

To start the engine, double click on *OpusCatMTEngine.exe* or run it from the command line. The next step is to download a pre-trained NMT model from the OPUS-MT repository by clicking on *Install OPUS model from Web*. Select the most recent model in your language pair (e.g., French-English) and click *Install model locally* to place it in your local AppData directory (e.g., C:\Users\Yuri Balashov\AppData\Local\opuscat). These two directories, whose contents are displayed in Figure 3 on page 35, are the only ones associated with OPUS-CAT on your computer, with the models and other data written to the second directory. No registry changes, and no external dependencies. Both directories can be safely deleted if the application is no longer needed. (I checked it.)

After installation, the model(s) will be displayed in the *Models* tab of the GUI (See Figure 2). You're now ready to click *Translate with model.* Now hold your breath: a state-of-the-art NMT system is running on your computer!

How Good Is the Base Model?

I ran some random tests with the base English-Russian model pre-trained on 50+ million parallel sentences from the OPUS corpora. I can report anecdotally (but in line with what Jost Zetzsche says in a recent post[4]) that the OPUS-CAT performance was roughly comparable to that of Google Translate or DeepL. For example, all three did equally well on sentences with center embedding, such as "The book the girl gave to the boy was interesting" (making the correct gender choices required in Russian) and "Yuri is easy to please" versus "Yuri is eager to please," which require different noun forms in Russian. And all three stumbled on really challenging (even for us!) "garden path" sentences, such as "The horse raced past the barn fell," generating fluent but inadequate output. DeepL did very well on the nested possessives in "My sister's favorite movie actor's largest car was green" (Самая большая машина любимого киноактера моей сестры была зеленого цвета), while OPUS-CAT missed "favorite" and "movie" (Самая большая машина актера моей сестры была зеленая), and Google inserted a disfluency in an otherwise acceptable translation (Самая большая машина у любимого киноактера моей сестры была зеленой). Not much should be read into such ad hoc evaluation. The online MT systems are constantly improving their output, but one can always set new traps for any engine to fall into. A more interesting question is the usefulness of MT systems for postediting and domain adaptation.

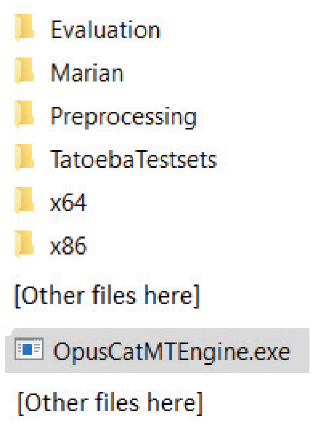
Fine-Tuning and Domain Adaptation in OPUS-CAT

Domain adaptation involves additional training of the base model with a relatively small amount of your golden data (e.g., 10–100K parallel sentences) in the form of translation memory eXchange (TMX) or parallel text files. To do this, click the *Fine-tune selected model* button (see Figure 2) and upload your TMX (Locally! You can shut down your internet if you're still worried) and choose a suitable model label. (See Figure 4 on page 36.)

Press *Fine-tune* and reopen the *Models* tab to track progress (see Figure 2). Fine-tuning may take several hours depending on the amount of data, but the engine will use only one CPU thread and 2048 MB of workspace by default, so you can safely do other things in parallel. Behind the scenes, OPUS-CAT will strip your TMX of all the tags and inline formatting, split it into a "training set" and a "validation set," and extract parallel Unicode text from both. The former is used to batch-train the model with conservative default hyperparameter settings (such as the "learning rate") to ensure a reasonable time frame, avoid overfitting (modeling the data too closely without learning the ability to generalize to new examples), and make your life easier. The validation set is used to check the model performance with BiLingual Evaluation Understudy (BLEU)[5] scores after every 100 weight updates. (See the blue line in Figure 5 on page 36.) In addition, the performance will

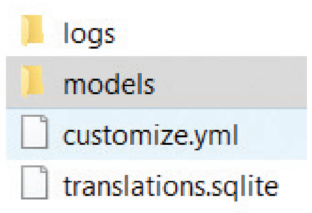
Figure 3: The contents of the two OPUS-CAT directories for version 1.1.0.3. They may look slightly different in later versions.

(a) C:\Users\Yuri Balashov\OpusCatMtEngine_v1.1.0.3\



Evaluation
Marian
Preprocessing
TatoebaTestsets
x64
x86
[Other files here]
OpusCatMTEngine.exe
[Other files here]

(b) C:\Users\Yuri Balashov\AppData\Local\opuscat\



logs
models
customize.yml
translations.sqlite

be scored against an "out-of-domain" validation set supplied with the base model (see the red line in Figure 5).

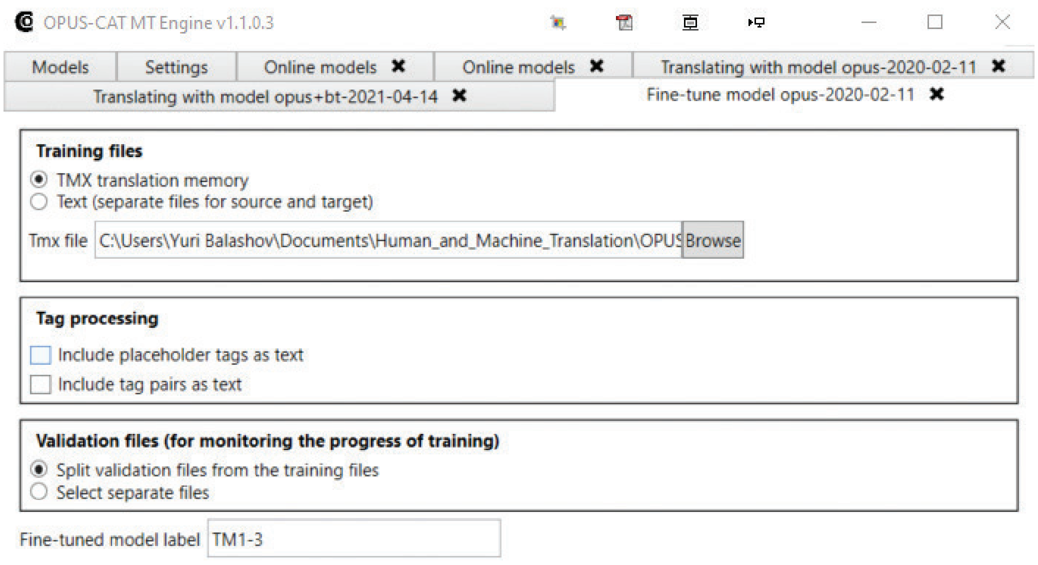
To access this chart, click *Show fine-tuning progress* when the fine-tuning is complete. If you're lucky (and your TM is good!) you should see your in-domain score improved by 10 or more BLEU points without overfitting and without degrading the out-of-domain performance by more than a few BLEU points. Otherwise, your model may start "hallucinating" the overlearned specialized content in translating generic sentences. ("Hallucination" is a term used in MT research to denote MT-generated content that has no counterparts in the source.)

Please be aware that each fine-tuning of the original model will create a separate subdirectory (about 1GB) in your local AppData directory or similar directory for your language pair. But any model can be easily removed by clicking *Delete selected model* from the GUI or directly. (Do not delete the base model!)

Customization and Evaluation (For More Advanced Users)

The quality of domain adaptation in any MT system is an open empirical question, and OPUS-CAT is no exception. You're in the best position to evaluate the output of a fine-tuned model, compare it with other customizable or generic MT systems, and decide whether it can be used in your work. You may be disappointed with the initial results, but don't give up. Experiment with different TMs, large and small. Try to increase the number of training "epochs" (i.e., complete runs through your fine-tuning data) by changing the *after-epochs* parameter from 1 to, say, 3 in the

Figure 4: Fine-tuning settings in OPUS-CAT


OPUS-CAT MT Engine v1.1.0.3

Models | Settings | Online models ✖ | Online models ✖ | Translating with model opus-2020-02-11 ✖

Translating with model opus+bt-2021-04-14 ✖ | Fine-tune model opus-2020-02-11 ✖

Training files

● TMX translation memory
○ Text (separate files for source and target)

Tmx file: C:\Users\Yuri Balashov\Documents\Human_and_Machine_Translation\OPUS [Browse]

Tag processing

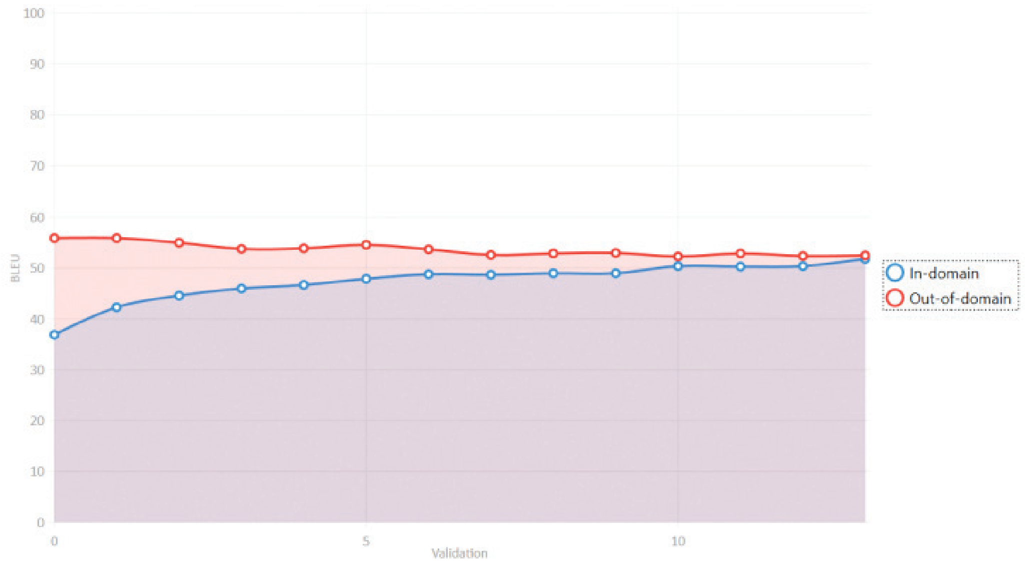
☐ Include placeholder tags as text
☐ Include tag pairs as text

Validation files (for monitoring the progress of training)

● Split validation files from the training files
○ Select separate files

Fine-tuned model label: TM1-3

Figure 5: Tracking fine-tuning progress with BLEU scores. From: Nieminen 2021



customize.yml file (see Figure 3b), which can be opened with Notepad++ or by clicking *Open fine-tune setting in text editor* in the *Settings* tab of the GUI (see Figure 2).

Consider using more demanding validation sets (e.g., your own domain-specific parallel Unicode text files with 100–300 sentence pairs) instead of letting the system split your TMX. And if you know what you're doing: try repeated training (with the same or different TMs) by replacing the model file in the base model directory with a similarly named file from the fine-tuned model directory and deleting the latter. As a last resort, try increasing the learning rate (roughly, the proportion by which the parameters of your model are adjusted after each training batch) from, for example, *learn-rate:0.00002* to, say, *learn-rate:0.00005* in the *customize.yml* file. But beware of overfitting: the system may effectively "memorize" your training data and become unable to generalize beyond them. This is a bit like manually oversteering a self-driving car into a wreck. If you have a good in-domain glossary, try to repeat the training with it and see what happens. (I haven't done it yet.) Finally, don't use any of your training sentences in validation or testing. That would run afoul of the first principle of machine learning! ("Never use your training data in testing!") I suggest using your CAT tool to filter out any TM matches ≥ 75% from both.

So far, my decidedly unscientific experimentation with all of that has produced mixed results. For example, using in-domain resources enabled the system to correct its initial translation of "fibrotic infiltration" from фибротическая инфильтрация to фибриозная инфильтрация. But the fine-tuned model sometimes misses or mistranslates entire fragments in the test sentences, such as "positive end expiratory pressure," while generating very fluent output. Post-editing it would definitely require extra care. Repeated training or increasing the number of epochs improves the translation of some parts of a test sentence but deteriorates others. Fine-tuning with a mammoth TM (general health care, about 350K units) and a more narrowly specialized custom validation set took over six hours, degraded both in- and out-of-domain performance, but did surprisingly well on that famous "garden path" sentence, translating "The horse raced past the barn fell" as Лошадь, которая бежала мимо сарая, упала ("The horse, which ran past the barn, fell"), and beating Google Translate, DeepL, and all my previous OPUS-CAT models. Neural networks are

For More Information

OPUS
https://opus.nlpl.eu
A growing collection of open multilingual corpora.

OPUS-MT
https://opus.nlpl.eu/Opus-MT/
A repository of 1000+ NMT models trained on OPUS. The training process is described here: *https://github.com/Helsinki-NLP/Opus-MT.*

OPUS-CAT MT Engine and CAT Plugins
https://helsinki-nlp.github.io/OPUS-CAT/
Documentation and installation instructions.

OPUS-CAT MT Engine and CAT Plugins
https://github.com/Helsinki-NLP/OPUS-CAT/issues
Lists known issues and debugging tips.

Marian NMT
https://marian-nmt.github.io

OpenNMT
https://opennmt.net

SacreBLEU
https://github.com/mjpost/sacrebleu
A robust implementation of BLEU scoring used in OPUS-CAT.

Subword NMT
https://github.com/rsennrich/subword-nmt
A subword segmentation algorithm based on byte-pair encoding and used in some older OPUS-CAT models.

SentencePiece
https://github.com/google/sentencepiece
A tokenization-free subword segmentation algorithm based on byte-pair encoding and unigram language model, used in most OPUS-CAT models.

generally inscrutable, but they can be awesome too.

Under the Hood

To those who are, like me, interested in how natural and artificial systems process bilingual content, OPUS-CAT offers a unique opportunity to take a peek under the hood without learning a lot of coding. You can explore the contents of some files in your model subdirectories with Notepad++. Some of these are configuration files while others are fine-tuning logs. For example, the *opus.spm32k-spm32k.vocab.yml* configuration file is quite special. It encodes a fixed bilingual vocabulary of about 60,000 entries used in all your models. Most entries are familiar words in both languages. Others are “subword pieces,” such as “al” and “gi,” which allow the model to learn the translation of rare or new words (including those unseen at training), such as “polymyalgia,” by mapping subword segments (e.g., “_poly my al gi a” to →поли ми ал ги я) and, hopefully, learning some morphology as well (e.g., “_patient s”→ _пациент ы).

The notion of “vocabulary” in NMT is, well, curious. You can visualize subword segmentation by checking the box next to your language pair in the *Translate with model* window. And you can play with the word alignment feature (if available for your model) to see which target words and subword segments correspond to which source words and their pieces, by hovering over the source sentence.

Good luck experimenting with OPUS-CAT! If you run into problems, make sure to check out the “debugging and known issues” GitHub site referenced in the sidebar. If you don’t find information on the problem you’re experiencing, consider posting a query on the site to see if anyone can offer a suggestion.[6] ATA

NOTES

1 Balashov, Yuri. “The Translator’s Extended Mind.” In *Minds and Machines* 30(2020), 349–83, *https://rdcu.be/b7r9T*; see also Zetzsche, Jost. “(More) Advanced Human–Computer Interaction for Translators,” *The ATA Chronicle* (January/February 2021), 30, *https://bit.ly/Zetzsche-Chronicle*.

2 For a recent discussion of this issue, see Zetzsche, Jost, “Data Privacy and MT Engines,” *The ATA Chronicle* (July/August 2021), 31, *https://bit.ly/data-privacy-MT*.

3 Nieminen, Tommi. “OPUS-CAT: Desktop NMT with CAT Integration and Local Fine-tuning.” In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (2021), 288–294, *https://bit.ly/Nieminen*.

4 Zetzsche, Jost. “Any Artistic Work, Especially One on a Large Scale,” *The Tool Box Journal*, Issue 21-4-324 (April 2021), *https://bit.ly/Zetzsche-ToolBox*.

5 BiLingual Evaluation Understudy (BLEU) score is an automatic metric for evaluating the quality of MT by comparing its output to reference translations produced by humans.

6 I thank Tommi Nieminen for numerous discussions and comments.

Yuri Balashov, CT is a professor of philosophy and a faculty fellow in the Institute for Artificial Intelligence at the University of Georgia. He is also a certified ATA member (English>Russian). He is currently working on a new project exploring cognitive, linguistic, and philosophical dimensions of human and machine translation (*www.yuribalashov.com*). balashov.yuri@gmail.com