TranslateLocally: Blazing-fast translation running on the local CPU

Nikolay Bogoychev and **Jelmer Van der Linde** and **Kenneth Heafield**
School of Informatics
University of Edinburgh
{N.Bogoych,Jelmer.vanderLinde,Kenneth.Heafield}@ed.ac.uk

Abstract

Every day, millions of people sacrifice their privacy and browsing habits in exchange for online machine translation. Companies and governments with confidentiality requirements often ban online translation or pay a premium to disable logging. To bring control back to the end user and demonstrate speed, we developed *translateLocally*. Running locally on a desktop or laptop CPU, *translateLocally* delivers cloud-like translation speed and quality even on 10 year old hardware. The open-source software is based on Marian and runs on Linux, Windows, and macOS.

1 Introduction

Neural Machine Translation (Bahdanau et al., 2015; Vaswani et al., 2017) is pervasive but has a reputation for high computational cost. The combination of the typically high computational cost, however, has pushed its delivery to the cloud, with a number of cloud providers available (Google, Microsoft, Facebook, Amazon, Baidu, etc.). Using a cloud based translation provider carries an inherent privacy risk, as users lose control of their data once it enters the web. Potential issues include public disclosure due to not understanding terms of service (Tomter et al., 2017), contractors reading user data (Lerman, 2019), use of user data for advertising, and data breaches.

To preserve privacy, we made a translation system that runs locally: *translateLocally*. Once a translation model is downloaded, it does not use an Internet connection. Running locally is challenging due to a number of factors: the model needs to be small enough to download on a user hardware; translation latency can't be hidden by splitting and parallelising the translation of a large documents across multiple machines; consumer hardware has highly variable computing power; availability of GPU computational resources can't be assumed.

We therefore focused on trimming model size and optimising speed for CPUs while aiming to preserve translation quality. The result is fast enough that users see translations update as they type with latency comparable to ping times to the cloud.

Targeting non-expert users, the open-source (primarily MIT) software[1] is also available as compiled binaries for Linux, Windows and Mac from the official webpage: https://translatelocally.com. Translation models for several language pairs are provided, while advanced users can add their own models.

2 Design

Our product is based on the Marian machine translation toolkit (Junczys-Dowmunt et al., 2018), heavily optimised for speed with a Qt based GUI.

2.1 Translation Engine

For the translation engine core, we used the same Marian fork as the one used by Bogoychev et al. (2020) for participating in the 2020 Workshop on Neural Generation and Translation's efficiency shared task (WNGT 2020, Heafield et al., 2020). We introduce binary lexical shortlists and streamlined binary model loading to the codebase, resulting in a comparable translation speed, but slightly faster loading time. We also add sentence splitting and formatting preservation are handled by a C++ wrapper around Marian.[2]

2.2 Translation Models

Our models are built with knowledge distillation (Kim and Rush, 2016), use lexical shortlists (Schwenk et al., 2007; Le et al., 2012; Devlin et al., 2014; Bogoychev et al., 2020) to reduce the size of the output layer, 8-bit integer arithmetic, and the

[1]https://github.com/XapaJIaMnu/translateLocally

[2]https://github.com/browsermt/bergamot-translator

Machine	Year	CPU	Cores	WPS
Laptop: Vaio PCG-41412L	2012	i5-2430M	2	1066
Desktop: iMac 27 inch	2012	i7-3770	4	3146
Desktop	2016	i7-6700	4	6548
Laptop: Dell XPS 9360	2017	i7-7500U	4	3378
Laptop: Dell Alienware 13R3	2017	i7-7700HQ	4	5888
Desktop	2019	AMD Ryzen 3600X	6	8791
Desktop	2019	i7-9700	8	9401
AWS c5.metal	2019	2x8275CL	48	70037

Table 1: Translation speed, in words per second (WPS), of the English→German model with 8-bit precision on various hardware. Translation used all cores. The table shows physical core count, not hyperthread count. WPS is averaged over 1M sentences. The timing measurement includes loading time but excludes sentence splitting, which was done in advance for this experiment.

simplified simple recurrent unit (Kim et al., 2019) for decoding.

We tested translation speed on a range of consumer hardware, shown in Table 1, using the million-sentence test set from the WNGT 2020 efficiency shared task and the *tiny11* preset English-German translation model from Bogoychev et al. (2020). This test set is already sentence split, so we did not include sentence splitting and format preservation in timing. Translating a million sentences provides ample opportunity to batch sentences of similar length and use all threads; users translating a few sentences will see slower throughput, but lower latency.

All of our student models available in the initial release are trained with the same *tiny11* configuration preset. Training, knowledge distillation and quantisation instructions are described in detail on github.[3] Users can follow those instructions to train, distil and quantise their custom models, achieving noticeable speedup over vanilla *float32* marian models, although any marian compatible models are supported in principle.

2.3 Language pairs

Our initial release includes 10 language pairs built for the Bergamot project (Table 2). We report average BLEU scores on WMT test sets up to WMT19 (Barrault et al., 2019), for all languages except for Icelandic and Norwegian. For Icelandic and Norwegian we report BLEU scores on self-crawled TED Talks test set, available on github.[4]

Languages pair	BLEU
en-es	35.0
es-en	35.3
en-et	25.1
et-en	30.8
cs-en	33.2
en-cs	25.9
en-de	41.8
is-en	23.7
nn-en	41.7
nb-en	42.7

Table 2: Language pairs and their BLEU scores in the initial release.

The models are distributed together with a lexical shortlist in an archive that is approximately 15MB in size. We are building and adding new models to the project.

2.4 GUI and user interaction

We chose the Qt[5] framework to build our graphical interface. The Qt framework is widely used, open source, free for non-commercial use and in active development. We support building against both Qt5 and Qt6, which allows us to support older Linux software distributions, like Ubuntu 16.04, which do not have easy access to Qt6 packages.

We took a minimalist approach the GUI, where the user is presented with a drop-down menu to

[3]https://github.com/browsermt/students/tree/master/train-student

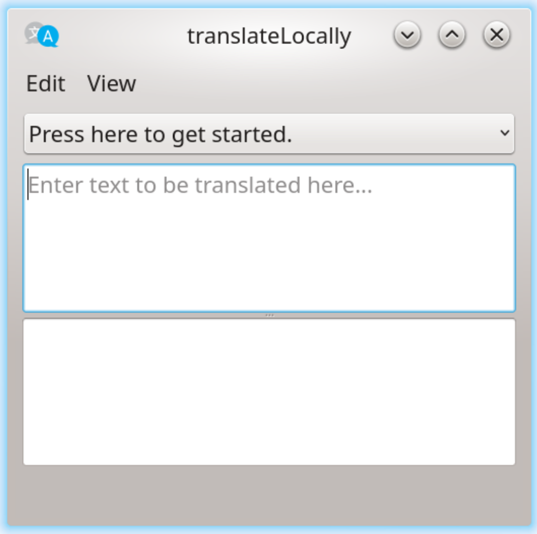
[4]https://github.com/browsermt/students/tree/master/isen/data

https://github.com/browsermt/students/tree/master/nnen/data

https://github.com/browsermt/students/tree/master/nben/data

[5]https://www.qt.io

select or download models, as well as a resizeable box where the user may input text. Translations will be shown underneath or besides the input text. Translations will start to appear as soon as the user begins inputting text. The view of the first run of the program is shown in Figure 1.



Press here to get started.

Enter text to be translated here...

Figure 1: First run view of *translateLocally*.

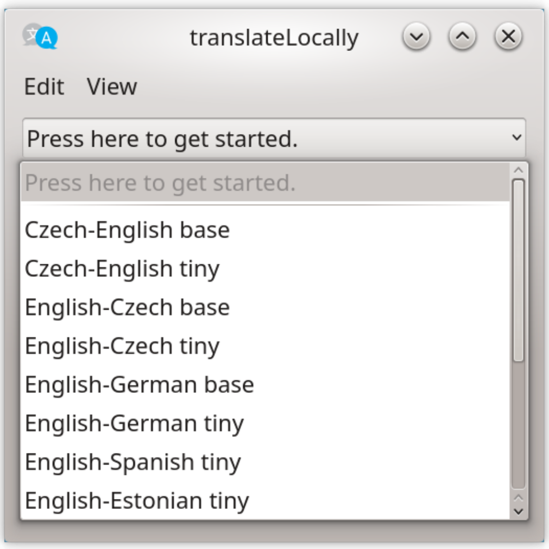
Downloading models from the Internet is done through a drop-down menu, as shown on Figure 2. In line with our privacy promise, the application only uses Internet access following explicit user action: to retrieve the list of available models and to download a new model. These downloads are static files. The HTTP request includes a user-agent field with the application version number. There is no cookie or other unique identifier. The directory containing downloaded models can also be copied to another machine to setup a system without Internet access; we are planning to ship a version with models included.

Once the model is downloaded, typing in the input box results in a translation, shown in Figure 3.

The application attempts to optimise thread count based on available cores and batch size based on available RAM though these can be overridden by the user, as shown on Figure 4. Fonts can also be changed through an OS-dependent dialog. Re-translating as a user types consumes power, so this feature can be disabled.

We also provide a model management screen where a user may delete downloaded models, or import custom models, as shown in Figure 5.

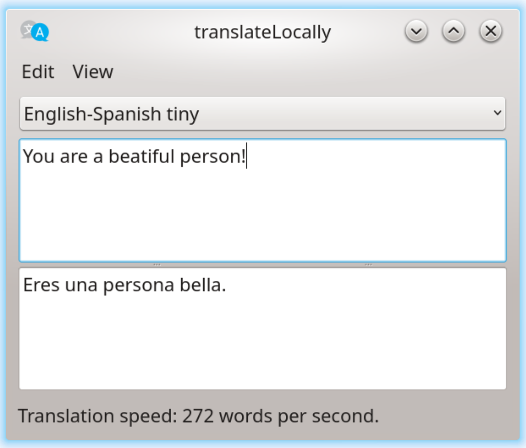
Our translation engine preserves whitespace be-



Press here to get started.

Press here to get started.
Czech-English base
Czech-English tiny
English-Czech base
English-Czech tiny
English-German base
English-German tiny
English-Spanish tiny
English-Estonian tiny

Figure 2: Select a model to download.



English-Spanish tiny

You are a beatiful person!

Eres una persona bella.

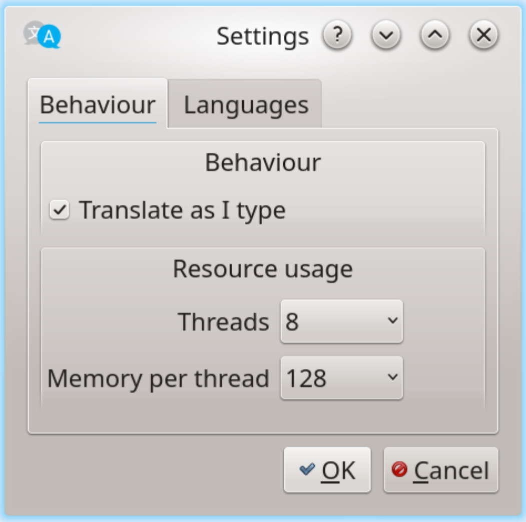
Translation speed: 272 words per second.

Figure 3: Translation view.

tween sentences, so users can copy/paste content and get a well formatted text, as shown on Figure 6, which also features the side-by-side view mode.

2.5 Distribution

Precompiled and packaged binaries for Windows, macOS and Ubuntu 20.04 are available on the official website. Users may fetch the source code from GitHub and build it on their local machines using *CMake*. The matrix multiplication library manually dispatches SSSE3, AVX2, AVX512BW and AVX512VNNI implementations based on CPUID. However, other kernels like activation functions are currently compiled without multiple versions and will be somewhat faster if compiled explicitly for a particular vectorised instruction set.



Behaviour

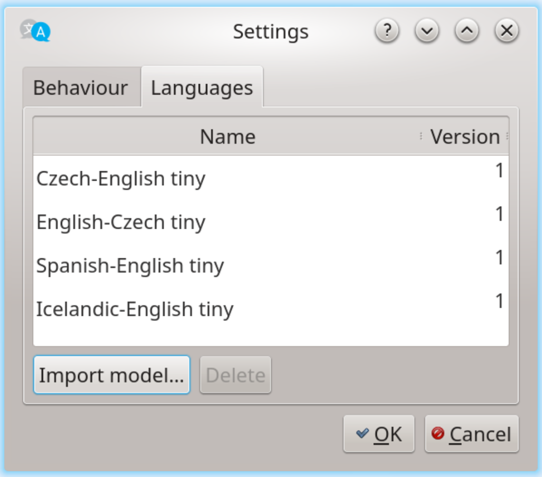
☑ Translate as I type

Resource usage

Threads: 8

Memory per thread: 128

Figure 4: Settings selection for the translation engine.



Name	Version
Czech-English tiny	1
English-Czech tiny	1
Spanish-English tiny	1
Icelandic-English tiny	1

Figure 5: Model management and import window.

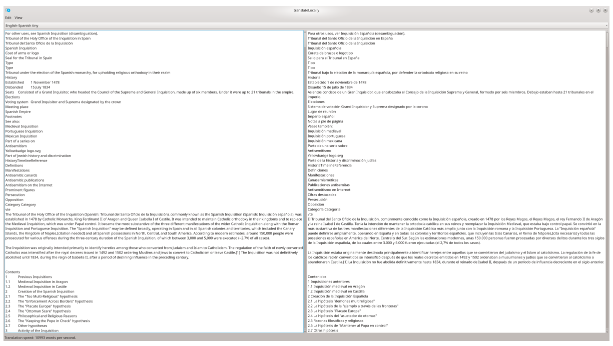


Figure 6: Translating a Big chunk of text from Wikipedia, with preservation of formatting.

3 Comparison against existing solutions

We compare against two existing desktop machine translation solutions: *Argos Translate*[6] and *OPUS-CAT MT Engine* (Nieminen, 2021). They both have slightly different use-cases and support different translation languages. We compare BLEU scores (Papineni et al., 2002) on a WMT19 test set (Barrault et al., 2019) for the English-German language pair, as well as wall-clock and CPU time. We measure only the time necessary for the actual translation. We ignore startup time and issue a translation of an unrelated text before running our test in order to discard any lazy initialisation time.

As only *translateLocally* supports all three platforms, we do pairwise comparison, once on Windows for OPUS-CAT vs *translateLocally*, and once on macOS for Argos Translate vs *translateLocally*.

3.1 Quality comparison

For the quality comparison we used the following models:

- For *translateLocally*, we used Bergamot's English-Germany tiny model[7] which is just 15 MB to download.
- For OPUS-CAT we used the English-German opus+bt-2021-04-13.zip[8] model, which is 275 MB in size.
- For Argos Translate we used their default English-German model which is downloaded through the UI, which is 87 MB in size when downloading.

We compare the BLEU scores on Table 3.

System	Model Size	BLEU
translateLocally	15 MB	41.8
OPUS CAT	275 MB	40.8
Argos Translate	87 MB	34.9

Table 3: BLEU score on WMT19 English-German as well as model sizes.

translateLocally's student architecture, coupled with 8bit integer model compression delivers the

[6]https://github.com/argosopentech/argos-translate

[7]http://data.statmt.org/bergamot/models/deen/ende.student.tiny11.tar.gz

[8]https://github.com/Helsinki-NLP/Tatoeba-Challenge/tree/master/models/eng-deu#opusbt-2021-04-13zip

smallest model size and the highest BLEU score. OPUS CAT has a comparable BLEU score, but the model is more than 15 times larger compared to *translateLocally*. Argos Translate has a much lower BLEU score than either of the two, and a model size that is right in the middle.

3.2 Argos Translate comparison

Argos Translate is based on OpenNMT and supports 13 language pairs, with more planned in the future.

Argos Translate is not fully cross-platform as there are no windows binaries provided. The developers do advertise that it is possible for users to self-build the product on Windows.

Finally, the macOS version is also available through the Apple app store, but it is paid,[9] whereas *translateLocally* is free.

We present our test results on Table 4. Both system were tested on a MacBook Pro 16" 2019, using 8 CPU threads to translate the whole WMT19 test set, which around 40k tokens. CPU time was measured using the Activity Monitor, and Words per second (WPS) is approximately calculated. Argos Translate does not allow the CPU threads to be configured by the user, so we matched the number of threads they use in *translateLocally*.

System	WPS	CPU Time	BLEU
translateLocally	7350	40s	41.8
Argos Translate	76	4378s	34.9

Table 4: *translateLocally* vs Argos Translate, translating 40k tokens for speed benchmark and BLEU scores on WMT19 English-German.

TranslateLocally is about 100 times faster and delivers vastly superior translation quality compared to Argos Translate.

3.3 OPUS-CAT comparison

Just like *translateLocally*, OPUS-CAT MT Engine (Nieminen, 2021) uses Marian as its translation engine. Unlike *translateLocally*, its translation engine is not optimised for speed. Furthermore the GUI is slow when handling large amounts of text. Simply pasting large chunks of text, such as the full "Crime and Punishment"[10] into OPUS-CAT takes nearly as long as *translateLocally* takes to paste *and translate* all the text.

The strength of OPUS-CAT comes from its plugins that integrate it with popular professional translator software, whereas our product does not support any CAT software.

OPUS-CAT has more language pairs available, which could also be used with *translateLocally*, but they are not optimised for speed.

Finally OPUS-CAT is only available for Windows, as it is build using the dot NET framework, whereas *translateLocally* is cross-platform.

For comparing OPUS-CAT vs *translateLocally*, we used a single threaded mode for both applications, as we found no way to force OPUS-CAT to use multiple threads, whether it is through their translation interface, or through their memoQ plugin.[11] We tested on a Windows Machine with 4 CPU core i9-9800H inside Parallels, measuring the CPU time from the task manager. We presplit the input of OPUS-CAT, as it doesn't have its own sentence splitter. Furthermore we excluded the copy/paste time from the OPUS-CAT measurements, as its XAML user interface is bad at handling large amounts of text. We present our results on Table 5.

System	WPS	CPU Time	BLEU
translateLocally	1250	34s	41.8
Opus-CAT	12	3363s	40.8

Table 5: *translateLocally* crippled to run single-threaded vs Opus-CAT, translating 40k tokens for speed benchmark and BLEU scores on WMT19 English-German.

Even with the added benefit of sentence-splitting and ignoring copy/paste time, and forcing single-threaded mode, OPUS-CAT is about 100 times slower than *translateLocally*.

4 Conclusion

We presented *translateLocally*, a desktop translation application, capable of high speed translations on a variety of hardware. Our software provides a viable alternative to cloud translation for users who are conscious of their privacy. Our product is 100 times faster than competing software and has none of the rate limitations of freemium cloud providers. We start with 10 high quality, optimised

[9] Free macOS version is distributed through pip.

[10] https://www.gutenberg.org/files/2554/2554-0.txt

[11] https://www.memoq.com

models and we aim to continuously add additional language pairs. As our product is open-source and cross-platform, it can be adopted by a wide range of users. The use of Marian as a translation engine allows for users to easily train their own models, potentially facilitating internal use for large organizations.

Acknowledgements

We thank all the researchers and engineers working on the Bergamot[12] project for making *translateLocally* possible, with special thanks to Ulrich Germann and Jerin Philip for their help with making the codebase cross platform, and to Graeme Nail for helping with deployment. We thank everyone who tested the beta version of the application and the reviewers for their comments and suggestions.

This work was conducted within the scope of the Research and Innovation Action *Bergamot*, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825303.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Nikolay Bogoychev, Roman Grundkiewicz, Alham Fikri Aji, Maximiliana Behnke, Kenneth Heafield, Sidharth Kashyap, Emmanouil-Ioannis Farsarakis, and Mateusz Chudyk. 2020. Edinburgh's submissions to the 2020 machine translation efficiency task. In *Proceedings of the The 4th Workshop on Neural Generation and Translation (WNGT 2020)*, Seattle.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Kenneth Heafield, Hiroaki Hayashi, Yusuke Oda, Ioannis Konstas, Andrew Finch, Graham Neubig, Xian Li, and Alexandra Birch. 2020. Findings of the fourth workshop on neural generation and translation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 1–9, Online. Association for Computational Linguistics.

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

Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Hai Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Rachel Lerman. 2019. Human workers can listen to google assistant recordings. *Associated Press*.

Tommi Nieminen. 2021. 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*, pages 288–294, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[12] `https://browser.mt/partners/`

Holger Schwenk, Marta R. Costa-jussà, and Jose A. R. Fonollosa. 2007. Smooth bilingual n-gram translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 430–438, Prague, Czech Republic. Association for Computational Linguistics.

Line Tomter, Martin H. W. Zondag, and Øyvind Bye Skille. 2017. Warning about translation web site: Passwords and contracts accessible on the internet. *Norsk Rikskringkasting AS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.