LitPC: A set of tools for building parallel corpora from literary works

Antoni Oliver
Universitat Oberta de Catalunya (UOC)
aoliverg@uoc.edu

Sergi Álvarez
Universitat Oberta de Catalunya (UOC)
salvarezvid@uoc.edu

Abstract

In this paper, we describe the LitPC toolkit, a variety of tools and methods designed for the quick and effective creation of parallel corpora derived from literary works. This toolkit can be a useful resource due to the scarcity of curated parallel texts for this domain. We also feature a case study describing the creation of a Russian-English parallel corpus based on the literary works by Leo Tolstoy. Furthermore, an augmented version of this corpus is used to both train and assess neural machine translation systems specifically adapted to the author's style.

1 Introduction

A parallel corpus is a collection of texts, each of which is translated into one or more other languages than the original. Parallel corpora are invaluable resources for researchers and professionals in the fields of literary studies, contrastive linguistics, machine translation, and literary translation. While there are many parallel corpora which can be accessed and checked online, the Opus Corpora[1] (Tiedemann, 2009) stands out as a primary collection. However, the representation of literary texts within these corpora is often limited. Opus Corpora lists few corpora for the literary domain and each corpus includes a limited number of parallel segments. Table 1 includes an overview of the total number of parallel segments in the Opus Corpora compared with the parallel segments for the literary domain, across three highly-resourced language pairs. The scarcity of resources in this domain means that researchers in literary studies, scholars in machine translation, and professional literary translators who seek to integrate parallel corpora for literary texts into their daily work must compile their own parallel corpora.

In this paper, we introduce different resources, software, and methodologies for the rapid and effective generation of parallel corpora from literary texts. While the programs and methodologies outlined are applicable across various subjects, the section dedicated to resources focuses specifically on literary texts.

Language Pair	All	Literature
eng-rus	185 M	17.6 K
eng-spa	922 M	97.1 K
eng-fra	787 M	0.1 M

Table 1: Parallel segments in Opus Corpora: total and literature corpora.

2 Sources for literary works

When compiling a literary corpus, the first thing we should consider is the copyright status of the original works and their translations. Copyright laws safeguard the rights of authors and translators for a specified duration-—typically ranging from 70 to 100 years, varying by country, after the death of the author or translator. Upon expiration of this term, the works enter into the public domain. Numerous online sources offer literary works for download; however, many such sources operate illegally, granting unauthorized access to copyrighted texts. To lawfully acquire copyrighted material, one must meticulously examine the copyright laws pertinent to each country. This process

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

often entails negotiations with the rights holders. It's noteworthy that in certain jurisdictions, purchasing a book may confer the right to utilize its content for non-commercial use, although this provision is not universally assured by all legal frameworks.

In this section, we explore various avenues for legally obtaining literary works that have entered the public domain, thereby ensuring unrestricted access and usage rights for any intended purpose.

To effectively develop machine translation (MT) systems focused on the literary domain, it is essential to create both parallel and monolingual corpora derived from literary texts with the language combinations involved in our project. There is a vast array of books which can be accessed online. Once found, they need to be downloaded, transformed into text format, and subsequently processed to generate the desired corpus. The following sources are particularly noteworthy:

- Project Gutenberg[2]
- Wikisource[3]
- Feedbooks[4]
- Hathi Trust Digital Library[5]
- Archive.org
- Standard Ebooks[6]

For our case study, we need to obtain the works by Leo Tolstoy in Russian and English. First of all, we use a website which is very popular for its extensive collection of Russian classical literature: Библиотека Максима Мошкова[7]. While it is ambiguous whether the site exclusively hosts public domain works, it is certain that Leo Tolstoy's contributions have been in the public domain for some time, as he passed away in 1910.

Additionally, to obtain the English translation, we use Project Gutenberg. This website offers a searchable database, complemented by daily updated catalog files. The LitPC toolkit encompasses a utility that interprets these catalogs in RDF format, enabling the search for books based on criteria such as author, title, language, the author's lifespan, and subject matter. These parameters can be specified as either strings or regular expressions, and searches can combine multiple criteria. Utilizing this functionality, the tool displays a curated list of works that align with the specified search parameters. Users have the flexibility to refine their search or modify the list of works. Upon finalization, the tool facilitates the download of the selected works in epub format to a designated directory.

[2]https://www.gutenberg.org/
[3]https://wikisource.org/
[4]https://www.feedbooks.com/catalog/public_domain
[5]https://www.hathitrust.org/
[6]https://standardebooks.org/
[7]http://lib.ru/

3 Basic processing steps

3.1 Conversion to text

After downloading all the works in the different language combinations, we need to process them. To do so, we need them in text format and in Unicode UTF-8 encoding. This process can be performed with any external tool, but for convenience, the LitPC toolkit provides a program capable of converting all epub files in a given directory to text. It can either place all the converted files in a given directory or produce a text file with all the contents of all epub files in the directory.

3.2 Segmentation

After conversion to text, the next step is text segmentation. This process divides the text into segments, which are usually sentences. This task can be performed by analyzing the periods that can be found in the text. A set of rules indicates whether each of the periods are splitting points. These rules are defined using regular expressions and are typically expressed using a standard XML format called Segmentation Rules eXchange (SRX). The toolkit includes a program that can use SRX files to segment a single file or all the files included in a directory. In the toolkit, different SRX files are included for different languages, even though other SRX files can also be used.

If no SRX file is available for a given language, we can use a trainable segmenter implemented in the NLTK library (Bird, 2006). This segmenter uses the algorithm created by Kiss and Strunk (2006) and can be trained on a large unsegmented corpus of a given language. The same corpus to be segmented can be used for training. Once trained, the segmenter can be customised using a list of abbreviations for the language. The LitPC toolkit also implements a program for training and customizing a segmenter and a program for segmenting using the trained model.

Both available segmenters can optionally add the paragraph mark (`<p>`) to the segmented text. As we will see in the next section, this mark can be useful for one of the automatic alignment strategies.

4 Automatic text alignment

After the basic pre-processing steps, we will have segmented text that we want to align. The segmented text can either include two text files or two directories–one for each language–containing the segmented text files.

There are several freely available automatic text alignment strategies. In the LitPC toolkit we use two different algorithms: one based on more classical techniques (which assumes parallel documents), and another one based on sentence embeddings (which is able to find translated segment pairs even in non-parallel texts).

As we are working with original literary works and its translations, it would seem clear that these are parallel documents for which we can seamlessly apply parallel document techniques. However, there are some elements which can turn a published original work into semi-parallel or comparable documents. Some of these factors include:

- The translated work is not from the same edition as the original work. In these cases, the changes are usually very small and can be handled by parallel document techniques. In other cases, however, the changes are significant. For example, a collection of some short stories is translated, but the translated published book changes the order of the short stories. In such cases, techniques for comparable corpora are appropriate.

- Either the original or the translation, or both, contain introductions, prefaces or other elements that make it impossible to align the document using classical techniques. The amount of human work required to manually edit the documents to make them equal is very important, so it is more efficient to use alignment techniques for comparable corpora. In the case of documents obtained from Project Gutenberg, they include a large section explaining the licence and terms of use.

- When bulk aligning documents, they must have the same name and be placed in two directories, or they must use the same name plus a suffix indicating the language of the document. This means spending additional time renaming all the files. To save time, in such cases we can align all the content in each language without taking into account the document information. This can be done with techniques for comparable documents.

4.1 From parallel documents

If we have two parallel documents or two directories, one containing a set of documents in a source language and the other containing the translated documents, we can use well-known techniques for document alignment. When working with a large number of documents in two directories, the relationship between the source and target documents must be easily deduced from their names. To facilitate this task, the source and target documents should have the same name, or only differ in the suffixes that indicate the language of the documents.

One of the most widely used automatic document alignment programs is Hunalign[8] (Varga et al., 2007). To achieve better results with this programme, we can:

- Include the paragraph mark (`<p>`) when segmenting the files.

- Use a bilingual dictionary for the language pair. The programme requires a bilingual dictionary to perform the alignment. Even though it is not mandatory and an empty file can be used, using bilingual dictionaries improves performance. Bilingual dictionaries are text files with one entry per line which follows the format *target word @ source word*, as in the following example for a English-Spanish alignment dictionary:

  ```
  hogar @ home
  ```

The toolkit provides Python scripts to create alignment dictionaries from the transfer dictionaries of the Apertium machine translation system (Forcada et al., 2011) and from MUSE[9] (*Multilingual Unsupervised and Supervised Embeddings*) (Conneau et al., 2017).

[8]https://github.com/danielvarga/hunalign
[9]https://github.com/facebookresearch/MUSE

4.2 From comparable documents

It is possible to find translated segments in a large collection of multilingual documents, even though they are not exact translated versions. If we can have a representation of each sentence in a document, we can then compare it to the representations which appear in another language to find the most similar one. If two sentences have sufficiently similar representations, we can infer that they are translation equivalents.

We can represent the sentences with sentence embeddings using a multilingual model. Then, calculating the cosine distance between all the sentences in the source language and in the target language, we can find those sentence pairs having the smaller distance. If this distance is small enough, we can select this sentence pair as translation equivalent. We have adapted an algorithm that can be found in the SBERT website, following the ideas of Artetxe and Schwenk (2019). The full process can be divided into the following steps:

- Representing all sentences in the source and target corpus by their sentence embeddings using a multilingual model. By default, as recommended by Reimers and Gurevych (2020), we use the LaBSE model (*Language-agnostic BERT Sentence Embeddings*) (Feng et al., 2022). To implement the algorithm, we use the Sentence-Transformers library[10], and LaBSE is integrated into the library. Any other model can be used with the provided algorithm.

- For each sentence in the source corpus, using its sentence embedding representation, the algorithm finds the k nearest neighbor sentences in the target corpus. Typical choices for k are between 4 and 16. The cosine distance between the embedding representations is used as a measure.

- All possible source-target sentence combinations are scored using a measure. Instead of directly using the cosine distance, a margin criterion is used, where the cosine distance for all the k nearest neighbors in both directions is considered, as explained by Artetxe and Schwenk (2019).

- The pairs with the highest margin scores are the most likely translated sentences. After the alignment, a visual inspection is required to set a minimum value and discard all pairs below that threshold. Usually, scores higher than 1.2 or 1.3 work very well.

[10]https://www.sbert.net/

This algorithm is implemented in the LitPC toolkit and can be used with or without a GPU unit.

5 Cleaning of parallel corpora

When obtaining an available parallel corpus or compiling our own corpus, we can find several common errors. Hence, it is always advisable to clean the corpus. The toolkit distributes a cleaning script that can perform, among others, the following cleaning operations:

- Apostrophe normalization: replacing the typographic apostrophe with the standard one.
- Removing HTML and XML tags.
- Replacing HTML/XML entities with their respective characters.
- Removing segment pairs where one is empty.
- Removing segments pairs where one or both are shorter than a given threshold.
- Removing segment pairs with equal segments.
- Removing segment pairs with a percent of numeric characters higher than a given threshold.
- We can set a file containing a set of strings. If a segment pair contains any of these strings, it will be removed.
- Removing segment pairs matching a set of regular expressions stated in a given file.
- Checking the source and target languages.
- Remove segments pairs with one or both segments written in upper case.
- Fixing encoding errors.

The Python langid library, which is able to detect 97 languages, is used to detect the language. However, the precision of language detection is relatively low for short text segments in a parallel corpus. Thus, a set of languages expected in the corpus can be given to increase the performance of the algorithm.

5.1 Rescoring of parallel corpora

Once cleaned, we have a corpus that does not include any of the aforementioned problems. To ensure these segment pairs are translation equivalents, we can calculate a score that provides a measure of the translation equivalence between the source and target segments in the segment pair. It can be achieved using sentence embeddings. We can encode the source and target segments in the segment pairs with sentence embeddings using a multilingual model, as explained in Section 4.2. The cosine distance between the source and target embeddings can be used as a score.

The toolkit provides a program that performs two actions:

- Language detection of the source and target segments, using fasttext[11] (Bojanowski et al., 2016). This tool offers two interesting features: it returns the detected language along with a detection confidence score and users can easily train their own language detection models.
- Scoring of all the segments with the cosine distance of the sentence embedding representation calculated using a multilingual model (LaBSE by default).

After the scoring process is completed, we provide a companion program used to select the parallel segments which match the language and the language detection confidence score. A minimum confidence score is required based on the cosine distance.

6 Enlarging the corpus using general language parallel corpora

In our use case, we utilize the compiled parallel corpus to train a neural machine translation system. Most likely, the compiled corpus will not be large enough to train an NMT system. Millions of parallel segments are required for a successful training. Preliminary experiments have shown that a good starting point would be 5 million segments, even though 10 or 15 million would be a better threshold.

In case there is a very large parallel corpus available for the required language pair, we can automatically select from the large general corpus the segments which are more similar to the ones found in the domain corpus. The procedure is very similar to the described in Moore and Lewis (2010). Let us call corpus A the small in-domain corpus (in our case, the parallel corpus from Tolstoy's works) and corpus B the very large general corpus. This process involves the following steps.

Fist of all, a language model is calculated from the source language part of corpus A. The perplexity of all source segments of corpus B is calculated using the language model. Then, all source and target segments of corpus B along with perplexity are stored in a database.

Once the calculations are finished, we select a given number of segments from the database, sorting them according to the perplexity in ascending order. This whole process can be performed using a program available in the LitPC toolkit.

7 Use case: Russian-English NMT model tailored to Tolstoy works

As an experimental part, we compiled a Russian-English parallel corpus from the works of Lev Nikolayevich Tolstoy, a Russian writer who was born in 1828 and died in 1919.

7.1 Original works and translations

We downloaded the original works and its translation into English in fb2 and epub format and converted them into text with Python scripts available in the LitPC toolkit.

We downloaded a total of 42 original works in Russian from Библиотека Максима Мошкова. We also downloaded all the English translations of Tolstoy's works available in Project Gutenberg.

The complete list of works used to create the corpus can be found in Appendix A. Please note that the list of original works is different from that of translated works.

After converting the files into text, they were segmented. For each language, all the segments of all the works were concatenated, and repeated segments were eliminated. As a result, we obtained a file containing all the unique Russian segments (a total of 118,755 segments) and a file containing all the unique English segments (a total of 200.013 segments).

7.2 Alignment

We used the alignment strategy for comparable corpora to align the two files containing unique segments in Russian and English. We can see these

[11] `https://fasttext.cc/`

two files as a comparable or semi-parallel corpus, as there is no line-by-line relationship between the two files. We obtained a file containing 74,998 aligned pairs. Each pair contains information on the margin score assigned to this pair. The file is sorted in descending order by the margin score; therefore, the first segments are more likely to be correctly aligned. In Table 2, we can observe some examples of alignments with higher scores (and correctly aligned) and with lower scores (incorrectly aligned).

7.3 Available Russian-English corpora

To build a large Russian corpus, we used a series of parallel corpora available in Opus Corpora. However, we must keep in mind that in the available corpora, we usually do not have information about the source and target languages. Any language in the pair can be the source, and both segments can be translated from another language. The following parallel corpora were used: CCMatrix, CCAligned, Wikimatrix, Paracrawl, and UNPC.

- **CCMatrix** (Schwenk et al., 2021b) is a parallel corpus extracted from web crawls. Each web document is converted to text, the language is identified and then segmented. All the segments in a given language are treated together, without having into account the document where it comes from, so the document information is not used. To create the parallel corpus from language A to language B, all segments in A are compared with all segments in B. The comparison is performed after converting the segments into sentence embeddings using a multilingual model. In this way, sentences in language A are very close in the multidimensional space to sentences in language B with similar meanings. Using the cosine distance, a margin score is calculated as explained in Artetxe and Schwenk (2019) to detect segment pairs with a high chance to be mutual translations.

- **CCAligned** is a parallel corpus compiled in a very similar manner as CCMatrix. In this case, though, document information is used. Only segments in the documents detected as parallel are aligned. The alignment is also performed using sentence embeddings. The process of creation of this corpus is described in El-Kishky et al. (2020).

- **Wikimatrix**: To compile this corpus (Schwenk et al., 2021a), Wikipedia articles in 85 languages were used. Authors don't limit the process to alignments with English and all possible language pairs are considered. It is very important to keep in mind that Wikipedia articles in different languages are different documents, and only eventually some articles or sections of articles are translations from other language versions. However, as the same articles in different languages explain the same concepts, it's likely to find segments being translation equivalents, even when the articles are written independently. To detect equivalent segments, similar techniques to the ones used in CCMatrix and CCAligned are used.

- **Paracrawl** (Bañón et al., 2020): A corpus developed within an EU-project that also provides tools for crawling the web in the search of parallel documents to be aligned. Initially, it only included EU languages, but more languages are being added, including the English-Russian pair.

- **UNPC**: The United Nations Parallel Corpus (Ziemski et al., 2016) was created from manually translated documents of the United Nations from 25 years (1990 to 2014). It is available in six official languages at the UN: Arabic, Chinese, English, French, Russian, and Spanish. All the alignments have been performed using Hunalign (Varga et al., 2007). The corpus does not contain information about the source language, but most of the original documents at UN are written in English or French. This is not a general corpus, but we have included it because it is a high-quality corpus.

7.4 Preprocessing of the available corpora

Before creating the corpus from the available corpora, we carried out several preprocessing steps:

- Elimination of repeated segments, using the standard Linux commands cat, sort, uniq and shuf.

- Cleaning of the corpora using the program described in section 5. The following cleaning operations have been performed:
  - Apostrophe normalization.

Source segment	Target segment	Margin score
Неприступная Мальта сдается без выстрела; самые неосторожные распоряжения увенчиваются успехом.	Impregnable Malta surrenders without a shot; his most reckless schemes are crowned with success.	1.6393
Берегись делать какое-нибудь различие, могущее нарушить равенство.	Beware of making any distinctions which may infringe equality.	1.6101
...	...	...
Запил, так запил!	When I drink, it's there!	1.0000
Скверность это, значит, не по закону это.	It's filthy, that's what I call it; it's not right.	1.0000

Table 2: Examples of obtained parallel segments from the Tolstoy's works and translations (the two segments with the highest and the lowest margin scores are presented)

- Removing of HMT/XML tags
- Removing of control characters
- Unescaping of HTML/XML entities
- Fixing encoding errors
- Removing segment pairs with one side empty
- Removing segments pairs with one part or both shorter than 10 characters
- Removing segment pairs with more thant 60% of numerical expression characters
- Removing segments pairs with equal source and target

Table 3 shows the size of the individual and final corpus after these operations.

Corpus	Size (segments)
CCMatrix	139,863,720
CCAligned	13,341,868
Wikimatrix	1,617,622
Paracrawl	5,318,501
UNPC	28,581,489
TOTAL	**178,686,030**

Table 3: Size of the Russian-English parallel corpora available in Opus Corpora used in the experiments

7.5 Rescoring of the corpora

Both the corpus from Tolstoy's works and the large corpus created from existing corpora were rescored using the tool described in Section 5.1. This rescoring process re-verifies the languages and computes a distance between the sentence embeddings of the source and target segments, using SBERT. The language detection model is able to return the language code together with a confidence score. In our experiments, we use a language detection threshold of 0.75 for both languages. This figure has been set after experimenting with several values and observing a good compromise between the final number of segments and the quality of the alignment. In Table 4 we can see the number of segments obtained after filtering out the segment pairs with a SBERT score lower than the indicated for the Tolstoy's parallel corpus. Table 5 shows the values for the large parallel corpus.

SBERT score	Segments
0.9	5,336
0.8	34,270
0.75	46,571
0.7	55,209
0.6	65,365
0.5	69,521
no filtering	74,998

Table 4: Size of the Tolstoy corpus with different minimum values of SBERT scores

7.6 Corpus combination

The size of Tolstoy corpus, regardless of the minimum SBERT score, and even without any filtering

SBERT score	Segments
0.9	31,899,731
0.8	116,247,528
0.75	135,595,366
0.7	145,935,126
0.6	154,980,096
0.5	158,014,261
no filtering	178,686,030

Table 5: Size of the large general corpus with different minimum values of SBERT scores

at all, is clearly insufficient to train an NMT system. In order to obtain a larger corpus, we used the corpus combination program described in Section 6. We have used the version filtered with a minimum value of SBERT score of 0.75. We have created three corpora in this way by selecting 10M, 20M and 30M parallel segments from the large corpus. Furthermore, we obtained three subcorpora for each size of the selected corpus:

- A training corpus using the selected segments from the large corpus and a fragment of the Tolstoy corpus (the remaining segments after the creation of the validation and evaluation corpus).
- A validation corpus using 5,000 segments from the Tolstoy corpus.
- An evaluation corpus using 5,000 segments from the Tolstoy corpus

As the parallel corpora have been deduplicated, no common segments are present in the three subsets. Table 6 shows the size of all the corpora:

	Segments		
Train	10,036,571	20,036,571	30,036,571
Val	5,000	5,000	5,000
Eval	5,000	5,000	5,000

Table 6: Size of the corpora used for training the NMT systems

7.7 Training of the NMT systems

The following NMT systems have been trained:

- A system using the large general corpus with a rescoring with a minimum SBERT score of 0.9 (Marian Gen.).
- A system using the corpus resulting from the combination of the Tolstoy corpus with 10M segments selected from the rescored general corpus (Marian 10M).
- A system using the corpus resulting from the combination of the Tolstoy corpus with 20M segments selected from the rescored general corpus (Marian 20M).
- A system using the corpus resulting from the combination of the Tolstoy corpus with 30M segments selected from the rescored general corpus (Marian 30M).

All the corpora were preprocessed using SentencePiece (Kudo and Richardson, 2018) with the following parameters: joining languages: False; model type: bpe; vocabulary size 64,000; vocabulary threshold: 50. The (sub)word alignments of the training corpus were computed calculated using eflomal (Östling and Tiedemann, 2016) in order to use guided-alignment in the training.

The NMT system was trained using the Mariannmt toolkit (Junczys-Dowmunt et al., 2018) with a transformer configuration. Two validation metrics were used: bleu-detok and cross-entropy. The early-stopping criterion was set to 5 on any of the metrics, and the validation frequency was set to 5,000.

7.8 Evaluation of the trained systems

We have evaluated all the trained systems and compared them with an open neural translation model (OpusMT[12]), that will be considered as the baseline, and two widely used commercial systems: Google Translate[13] and DeepL[14]. To evaluate the systems we used three automatic metrics implemented in Sacrebleu[15] (Post, 2018): BLEU, chrF2 and TER. Appendix B shows the signatures of the three metrics stating the exact configuration parameters as reported by Sacrebleu.

Table 7 shows the evaluation results. In the evaluation, paired bootstrap resampling test with 1,000 resampling trials have been performed, using the -paired-bs option in Sacrebleu. In this way, each system is pairwise compared to the baseline system OpusMT. Assuming a significance threshold

[12]`https://github.com/Helsinki-NLP/OPUS-MT-train/tree/master/models/ru-en`
[13]`https://translate.google.com/`
[14]`https://www.deepl.com`
[15]`https://github.com/mjpost/sacrebleu`

System	BLEU (μ ± 95% CI)	chrF2 (μ ± 95% CI)	TER (μ ± 95% CI)
Baseline: OpusMT	17.8 (17.8 ± 1.0)	42.4 (42.4 ± 0.8)	68.8 (68.8 ± 1.2)
MarianGen.en	16.0 (16.0 ± 0.8) (p = 0.0010)*	40.8 (40.8 ± 0.7) (p = 0.0010)*	70.0 (70.0 ± 1.1) (p = 0.0060)*
Marian 10M	18.7 (18.7 ± 0.9) (p = 0.0240)*	42.2 (42.2 ± 0.8) (p = 0.1688)	67.6 (67.6 ± 1.1) (p = 0.0200)*
Marian 20M	19.2 (19.2 ± 0.8) (p = 0.0020)*	43.7 (43.7 ± 0.7) (p = 0.0010)*	67.2 (67.2 ± 1.0) (p = 0.0020)*
Marian 30M	19.1 (19.1 ± 0.8) (p = 0.0040)*	43.2 (43.2 ± 0.7) (p = 0.0120)*	67.7 (67.7 ± 1.1) (p = 0.0150)*
GoogleT.en	25.6 (25.6 ± 1.0) (p = 0.0010)*	50.3 (50.3 ± 0.8) (p = 0.0010)*	61.6 (61.6 ± 1.2) (p = 0.0010)*
DeepL	24.9 (24.9 ± 1.1) (p = 0.0010)*	49.7 (49.7 ± 0.8) (p = 0.0010)*	63.1 (63.1 ± 1.3) (p = 0.0010)*

Table 7: Evaluation results for the NMT systems

of 0.05, the null hypothesis can be rejected for p-values < 0.05 (marked with ”*” in the tables.)

Regarding the BLEU score, all systems except the Marian Generic get better results than Opus MT. Even the Marian 10M improves compared the baseline system for this metric. In fact, almost all tailored Marian systems are obtaining better results than the baseline OpusMT for all metrics. The only exception is chrF2 score for Marian 10M, that obtains slightly lower results than the baseline, but falling to pass the significance test.

This leads us to conclude that the author-tailoring methodology outlined in this paper can be highly productive. This assertion is further supported by comparing the evaluation metrics of all the tailored systems with those of the Marian Generic systems, which were trained using the same parameters as the tailored systems.

Nevertheless, it’s important to acknowledge that both the baseline system and all the trained systems achieve evaluation scores which are lower to those of commercial systems. This suggests that there is still room for improvement, both in the selection of general and literature-specific corpora, as well as in improving the training processes. Anyway, it’s important to note that the training sets of the commercial systems may include segments in our evaluation set and this could lead to over-optimistic evaluations.

8 Conclusions and future work

In this paper, we introduce LitPC, a toolkit designed for swiftly generating parallel corpora from literary texts. These versatile tools can also be applied to create parallel corpora for various other subjects. All tools are made available under a free license (GNU-GPL v.3) and can be downloaded from GitHub[16].

We have additionally showcased an experiment involving the development of author-tailored Russian-English NMT systems for Tolstoy’s works. The evaluation demonstrates the efficacy of the proposed methodology, although there remains potential for further enhancements to attain results comparable to those of the examined commercial systems. In future experiments we plan to fine-tune exiting models instead of training from scratch and comparing the two strategies.

The future work is planned in two directions: to experiment with fine tuning existing NMT models for literature; and to explore the use of parallel corpora aligned in larger units than segments, as paragraphs or chapters, as suggested by Voigt and Jurafsky (2012).

References

Artetxe, Mikel and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transac-*

[16]`https://github.com/aoliverg/litpc`

tions of the association for computational linguistics, 7:597–610.

Bañón, Marta, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, et al. 2020. Paracrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567.

Bird, Steven. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Conneau, Alexis, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Feng, Fangxiaoyu, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891.

Forcada, Mikel L, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25:127–144.

Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, et al. 2018. Marian: Fast neural machine translation in c++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121.

Kiss, Tibor and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational linguistics*, 32(4):485–525.

Moore, Robert C and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224.

Östling, Robert and Jörg Tiedemann. 2016. Efficient word alignment with markov chain monte carlo. *The Prague Bulletin of Mathematical Linguistics*, (106):125–146.

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

Reimers, Nils and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525.

Schwenk, Holger, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021a. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361.

Schwenk, Holger, Guillaume Wenzek, Sergey Edunov, Édouard Grave, Armand Joulin, and Angela Fan. 2021b. Ccmatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500.

Tiedemann, Jörg, 2009. *News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces*, volume V, pages 237–248.

Varga, Dániel, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. In *Recent Advances in Natural Language Processing IV*, pages 247–258. John Benjamins.

Voigt, Rob and Dan Jurafsky. 2012. Towards a literary machine translation: The role of referential cohesion. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 18–25.

Ziemski, Michał, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534.

Appendix A. List of Tolstoy's works and translations

Original Tolstoy's works:

Детство; Отрочество; Юность; Семейное счастье; Война и мир; Анна Каренина; Воскресение; Два гусара; Альберт; Поликушка; Холстомер; Смерть Ивана Ильича; Дьявол; Казаки; Набег; Рубка леса; Записки маркёра; Утро помещика; Метель; Разжалованный; Три смерти; Крейцерова соната; Отец Сергий; Хаджи-Мурат; Севастополь в декабре месяце; Севастополь в мае Севастополь в августе 1855 года; Хозяин и работник; Алеша Горшок; Ягоды; Корней Васильев; Отец Сергий (варианты); Сказки; Декабристы; Первый винокур, Власть тьмы;

Записки сумасшедшего; Божеское и человеческое

English translations:

A Letter to a Hindu; Anna Karenina; A Russian Proprietor, and Other Stories; Bethink Yourselves!"; Boyhood; Childhood; Fables for Children, Stories for Children, Natural Science Stories, Popular Education, Decembrists, Moral Tales; Father Sergius; Fruits of Culture; Katia; Master and Man; My Religion; On the Significance of Science and Art; Plays: Complete Edition, Including the Posthumous Plays; Redemption and two other plays; Resurrection; Sebastopol; Sevastopol; The Awakening (The Resurrection); The Cause of it All; The Census; in Moscow; The Cossacks: A Tale of 1852; The Devil; The First Distiller; The Forged Coupon, and Other Stories; The Invaders, and Other; Stories; The Journal of Leo Tolstoi (First Volume—1895-1899); The Kingdom of God is Within You Christianity and Patriotism Miscellanies; The Kingdom of God Is Within You" Christianity Not as a Mystic Religion but as a New Theory of Life; The Kingdom of God is Within You; What is Art?; The Kreutzer Sonata and Other Stories; The Light Shines in Darkness; The Live Corpse; The Power of Darkness; Three Days in the Village, and Other Sketches. Written from September 1909 to July 1910.; Tolstoi for the young: Select tales from Tolstoi; Tolstoy on Shakespeare: A Critical Essay on Shakespeare; War and Peace, Book 01: 1805; War and Peace; What Is Art?; What Men Live By, and Other Tales; What Shall We Do?; What to Do? Thoughts Evoked by the Census of Moscow; What to Do? Thoughts Evoked By the Census of Moscow; Where Love is There God is Also; Youth;

Apendix B. Metric signatures

- BLEU nrefs:1 | bs:1000 | seed:12345 | case:mixed | eff:no | tok:13a | smooth:exp | version:2.3.1
- chrF2 nrefs:1 | bs:1000 | seed:12345 | case:mixed | eff:yes | nc:6 | nw:0 | space:no | version:2.3.1
- TER nrefs:1 | bs:1000 | seed:12345 | case:lc | tok:tercom | norm:no | punct:yes | asian:no | version:2.3.1