The OPUS Resource Repository: An Open Package for Creating Parallel Corpora and Machine Translation Services

Mikko Aulamo, Jörg Tiedemann
Department of Digital Humanities / HELDIG
University of Helsinki, Finland
{name.surname}@helsinki.fi

Abstract

This paper presents a flexible and powerful system for creating parallel corpora and for running neural machine translation services. Our package provides a scalable data repository backend that offers transparent data pre-processing pipelines and automatic alignment procedures that facilitate the compilation of extensive parallel data sets from a variety of sources. Moreover, we develop a web-based interface that constitutes an intuitive frontend for end-users of the platform. The whole system can easily be distributed over virtual machines and implements a sophisticated permission system with secure connections and a flexible database for storing arbitrary metadata. Furthermore, we also provide an interface for neural machine translation that can run as a service on virtual machines, which also incorporates a connection to the data repository software.

1 Introduction

Parallel corpora are tremendously useful for a variety of tasks. Their natural home is the development of machine translation (MT) where data-driven approaches such as neural MT are data-hungry and still most language pairs and textual domains are under-resourced. Besides MT, there is also plenty of other work that exploits parallel corpora for, e.g., annotation projection (Tiedemann and Agić, 2016), representation learning (Artetxe and Schwenk, 2018), word sense disambiguation (Lefever, 2012), discovery of idiomatic expressions (Villada Moirón and Tiedemann, 2006) and automatic paraphrase detection (Sjöblom et al., 2018). Finally, we should not forget translation studies (Doval and Sánchez Nieto, 2019) and computer-aided language learning (Frankenberg-Garcia, 2005) as additional application areas.

We have a long tradition in collecting and providing parallel corpora for the public use. OPUS[1] has become the major hub for such data sets and we are now in the process of developing software that makes it easier for external collaborators to contribute to the collection. For this purpose, we have created the OPUS resource repository toolkit that we introduce in this paper. The purpose of this software package is to implement a scalable data processing pipeline that can be accessed via intuitive interfaces and powerful and secure APIs.

Figure 1 illustrates the overall architecture of the repository software. The package is divided into a distributed backend that combines storage servers, metadata databases, a cluster of preprocessing nodes, and a frontend that provides the interface to the backend via secure HTTPS connections. More details about both parts will be given further down.

Finally, we also implement a translation tool that connects to the repository software. The main purpose of that tool is to serve translation engines that can be trained on parallel data from the repository or other sources via a clean web-interface with options for donating data to the project. More details are given in section 3.2.

The software itself is available as open source from github[2] and we provide a public instance of the toolkit from `http://opus-repository.ling.helsinki.fi/`. The implementation of the online translator is also available[3] and currently we run an instance for the translation between Scandinavian languages (Swedish, Danish, Norwegian) and Finnish.[4]

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

[2] https://github.com/Helsinki-NLP/OPUS-repository

[3] https://github.com/Helsinki-NLP/OPUS-translator

[4] https://translate.ling.helsinki.fi

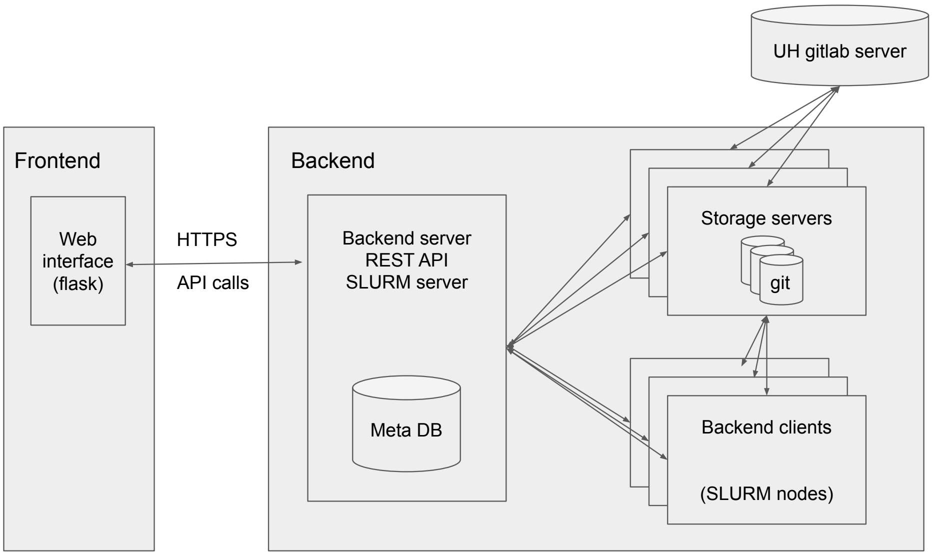


Figure 1: Overall architecture of the OPUS resource repository software.

2 Resource Repository Backend

The backend of the resource repository is based on software development for LetsMT! (Vasiļjevs et al., 2012), a project within the ICT Policy Support Programme of the European Commission.[5] The basic architecture of the OPUS resource repository is the same as in the package developed in that project but the software has been updated and extended in various ways:

- The job scheduler now uses SLURM[6] for workload management and the distribution of jobs over connected nodes in the cluster.
- The storage servers rely on git as their default data repository backend. Other backends are also still supported such as SVN repositories and plain file systems without revision control. We also support the connection to a remote git server for automatic replication and backups of the data.
- The software has been updated to run on Ubuntu servers with the current versions of software libraries and tools. This update included numerous bug fixes and performance optimizations to reduce bottlenecks and memory leaks in the backend.
- The data processing pipeline has been improved in various ways, e.g. integrating modern language identifiers (langid.py (Lui and Baldwin, 2012) and CLD2[7]) and robust document conversion tools such as ApacheTika[8] running in server mode.
- The APIs have been extended with many additional functionalities. This includes changes to the job control API, metadata search and the storage API. We now also support the creation of translation memories for better interoperability. More details can be found in the online documentation of the repository software at `https://github.com/Helsinki-NLP/OPUS-repository/tree/master/doc`
- New sentence alignment modes have been added, word alignment using eflomal[9] (Östling and Tiedemann, 2016) has been integrated and an experimental call for setting up interactive sentence alignment has been added.

An important feature for the backend is scalability. The system has been designed in a modular way to ensure that additional servers can be connected to the network to adjust for increasing workloads. Figure 1 shows the overall picture of the backend architecture. The main server provides the REST API that can be accessed from the outside for the different actions and requests

[5] http://project.letsmt.eu

[6] https://slurm.schedmd.com

[7] https://github.com/CLD2Owners/cld2

[8] https://tika.apache.org

[9] https://github.com/robertostling/eflomal

to be send to the system. It also serves the metadata DB that stores the essential information for all data records, users and permissions. The actual data sets can be distributed over several storage servers. In the basic setup, they will also be placed on the main backend server with local git repository on mounted file systems. Communication between all nodes in the backend and from the frontend to the backend is done via secure HTTPS connections with signed certificates and private keys. The main bottleneck for the repository is data pre-processing and alignment. For scalability and robustness we, therefore, implement a workload manager based on SLURM that can distribute data processing tasks to various worker clients in the backend cluster. Those workers communicate with the SLURM server and with the storage and metadata servers via the repository API.

The metadata DB is based on a flexible key-value store using TokyoTyrant and TokyoCabinet[10]. It enables fast access and complex queries about data records and configurations. It scales well to large numbers of data records and provides the essential functionality that we require in a system, which will be further enhanced in the future and that requires extensive meta-information, which is not pre-defined and strictly categorical. The key-value store allows arbitrary data records to be connected with any data record in the repository. It is also used to control jobs and process configurations.

Each backend client includes the software necessary for converting and processing data including language identification, data validation, text extraction, sentence boundary detection, tokenization and sentence alignment. The result of this conversion process is a unified format for parallel corpora based on XCES Align for the standoff sentence alignment and a standalone XML format for encoding the textual documents. With this, we follow the structure of OPUS to be immediately compatible with that data source.

The repository software and pre-processing pipelines received substantial improvements. The system now runs ApacheTika servers for robust document conversion, another server for language identification based on langid.py and CLD2 and the system supports pre-processing with UD-compatible models using UDpipe (Straka and Straková, 2017) with pre-trained models from the universaldependencies project.[11] In the basic setup, this includes sentence boundary detection and tokenisation but even full parsing is supported. Alternative pre-processing tools are also available such as the Moses tokenizers (Koehn et al., 2007) and OpenNLP pre-processing modules.[12] Additional tools may be added later on.

A final feature is the automatic word aligner based on eflomal and pre-trained models from the OPUS project. The system can use model priors derived from existing OPUS data to reliably align even the smallest document pair that arrives in the repository. This is, however, an experimental feature and not enabled by default.

The backend provides a number of complex APIs that control the system. Those APIs are only accessible via verified connections. The frontend implements the public interface that allows external users to communicate with the system. This interface supports the essential functionality of the system. Details will be given in the following section.

3 Interfaces

The project includes open source online interfaces for the resource repository backend[13] and for a translation system.[14] Both of the interfaces are written in Python using the Flask web framework[15] and our running instances are accessible via any common web browser. Users can register on either website and the same account may be used to login to both of these services. Using the translator is possible without a user account, but gaining access to the repository requires being logged in as a registered user.

3.1 Online repository

The online repository website is a graphical user interface for the resource repository API. With the API, one can upload documents, that are translations of each other, and the backend aligns them on the sentence level. The interface enables the use of the API without certification and command line operations. In practice, whenever a user takes an action on the repository website, a command line request is sent to the API. The API sends a response, which is parsed and displayed on the web-

[10] https://fallabs.com/tokyotyrant/

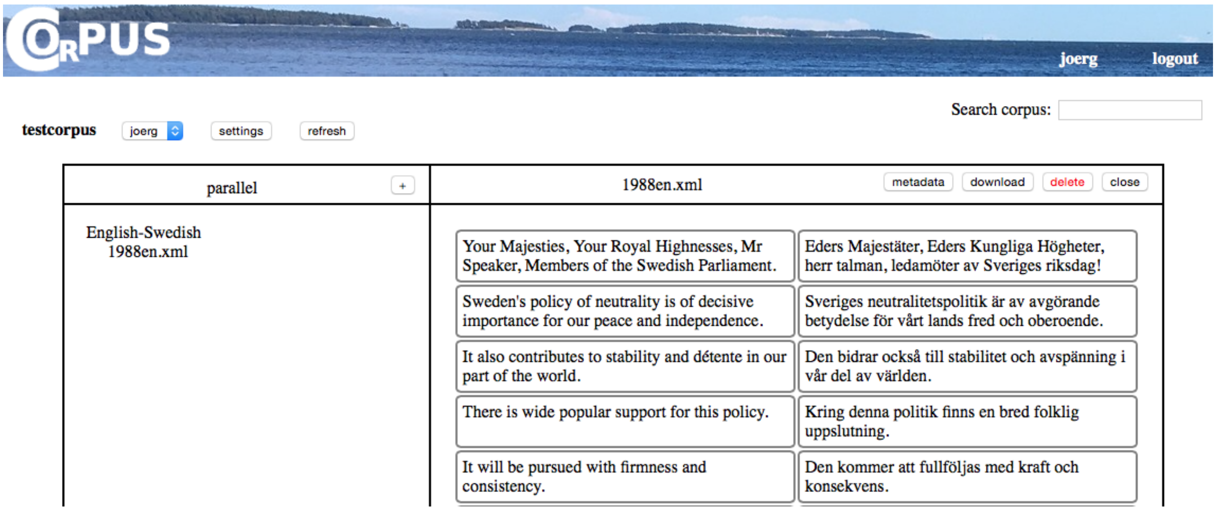
[11] https://universaldependencies.org

[12] https://opennlp.apache.org

[13] https://github.com/Helsinki-NLP/OPUS-interface

[14] https://github.com/Helsinki-NLP/OPUS-translator

[15] http://flask.pocoo.org/



Your Majesties, Your Royal Highnesses, Mr Speaker, Members of the Swedish Parliament.	Eders Majestäter, Eders Kungliga Högheter, herr talman, ledamöter av Sveriges riksdag!
Sweden's policy of neutrality is of decisive importance for our peace and independence.	Sveriges neutralitetspolitik är av avgörande betydelse för vårt lands fred och oberoende.
It also contributes to stability and détente in our part of the world.	Den bidrar också till stabilitet och avspänning i vår del av världen.
There is wide popular support for this policy.	Kring denna politik finns en bred folklig uppslutning.
It will be pursued with firmness and consistency.	Den kommer att fullföljas med kraft och konsekvens.

Figure 2: A screenshot of the OPUS resource repository interface.

site in an appropriate way depending on the type of request. Anytime a web page or a part of a web page is generated, all the data, that is presented, is received from the API, e.g. lists of corpora, documents, jobs or users.

In order to use the interface, one must first register to the website and login. Once logged in, users may create new corpora with metadata and settings, which can be edited later. User groups can also be created, and a corpus may be set to be accessible to only a specific group. Users can upload translated documents to a corpus, which are then aligned in the backend. Currently, the allowed document formats are PDF, DOC, TXT, XML, HTML and EPUB. Multiple files may be uploaded at once using TAR, TAR.GZ, or ZIP archives. All uploaded documents and the resulting alignment files are browsable using the tree file system on the interface. Figure 2 shows an example. The website also has a function to search for public corpora and to clone them for further use.

3.2 Online translator

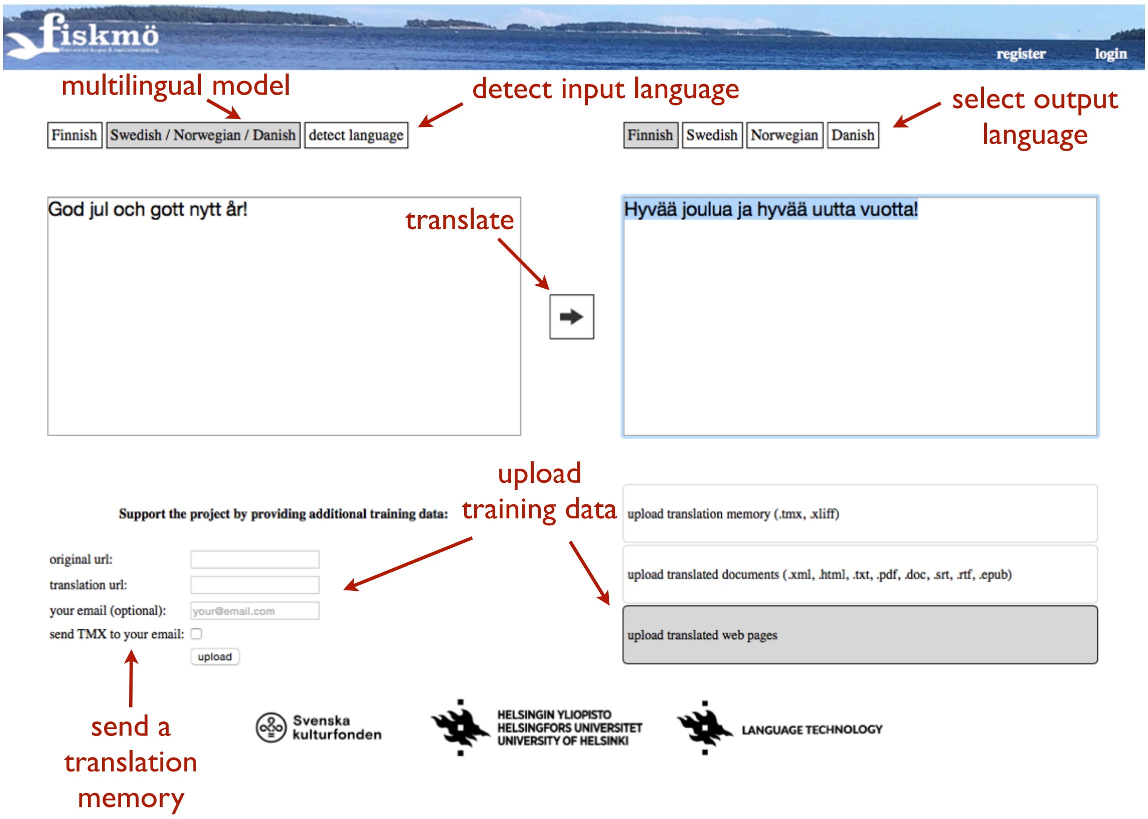
The current translation application runs two multilingual translation models: Finnish to Danish/Norwegian/Swedish (fi-da/no/sv) and Danish/Norwegian/Swedish to Finnish (da/no/sv-fi). The models are trained using the Marian Neural Machine Translation framework (Junczys-Dowmunt et al., 2018) and they run using the framework's web-socket server feature. To translate a text, a source language is first selected from two options: Finnish or Danish/Norwegian/Swedish. The source language can also be automatically detected. Language detection is performed using pycld2 Python bindings[16] for Google Chromium's Compact Language Detector 2.[17] The target language is chosen from Finnish, Danish, Norwegian, or Swedish. Once the source and target languages are selected and an input text is entered, the text can be translated. If the source language is Finnish and the target language is either Danish, Norwegian or Swedish, the source sentence is translated with fi-da/no/sv model. If the source language is Danish/Norwegian/Swedish and the target language is Finnish, da/no/sv-fi model is used. The resulting translation is represented on the web page. A screenshot of the interface is shown in Figure 3.

The online translator includes a feature to donate more training data. There are three different options to upload data. The first option is to upload translation memories, which can be either TMX or XLIFF files. The second option is to upload documents that are translations of each other and the files must be in XML, HTML, TXT, PDF, DOC, SRT, RTF or EPUB format. In the third option, the user enters two URLs, which point to two web pages that are translations of each other. When uploading translated files or entering translated web pages, the user has an option to receive a TMX file created from their contributed data.

4 Conclusions

This paper presents a new public resource repository for creating and managing parallel corpora

[16] https://pypi.org/project/pycld2/
[17] https://github.com/CLD2Owners/cld2


fiskmö
register
login
Finnish
Swedish / Norwegian / Danish
detect language
Finnish
Swedish
Norwegian
Danish
upload
upload translation memory (.tmx, .xliff)
upload translated documents (.xml, .html, .txt, .pdf, .doc, .srt, .rtf, .epub)
upload translated web pages
Svenska kulturfonden
HELSINGIN YLIOPISTO HELSINGFORS UNIVERSITET UNIVERSITY OF HELSINKI
LANGUAGE TECHNOLOGY

God jul och gott nytt år!

Hyvää joulua ja hyvää uutta vuotta!

Support the project by providing additional training data:

original url:

translation url:

your email (optional): your@email.com

send TMX to your email: □

multilingual model

detect input language

select output language

translate

upload training data

send a translation memory

Figure 3: A screenshot of the translator interface.

with a scalable backend and intuitive interfaces. A translation demonstrator is also provided and the software is released as open source.

Acknowledgments

The work was supported by the Swedish Culture Foundation and we are grateful for the resources provided by the Finnish IT Center for Science, CSC.

References

Mikel Artetxe and Holger Schwenk. 2018. http://arxiv.org/abs/1812.10464 Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *CoRR*, abs/1812.10464.

Irene Doval and M. Teresa Sánchez Nieto. 2019. https://doi.org/https://doi.org/10.1075/scl.90 *Parallel Corpora for Contrastive and Translation Studies – New resources and applications*. John Benjamins.

Ana Frankenberg-Garcia. 2005. Pedagogical uses of monolingual and parallel concordances. *ELT journal*, 59(3):189–198.

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. http://www.aclweb.org/anthology/P18-4020 Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. http://dl.acm.org/citation.cfm?id=1557769.1557821 Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Els Lefever. 2012. *ParaSense: parallel corpora for word sense disambiguation*. Ph.D. thesis, Ghent University.

Marco Lui and Timothy Baldwin. 2012. https://www.aclweb.org/anthology/P12-3005 langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. Association for Computational Linguistics.

Eetu Sjöblom, Mathias Creutz, and Mikko Aulamo. 2018. http://arxiv.org/abs/1809.07978 Paraphrase detection on noisy subtitles in six languages. *CoRR*, abs/1809.07978.

Milan Straka and Jana Straková. 2017. http://www.aclweb.org/anthology/K/K17/K17-3009.pdf Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jörg Tiedemann and Željko Agić. 2016. https://doi.org/doi:10.1613/jair.4785 Synthetic treebanking for cross-lingual dependency parsing. *Journal of Artificial Intelligence Research*, 55:209–248.

Andrejs Vasiļjevs, Raivis Skadiņš, and Jörg Tiedemann. 2012. http://www.aclweb.org/anthology/P12-3008 LetsMT!: Cloud-based platform for do-it-yourself machine translation. In *Proceedings of the ACL 2012 System Demonstrations*, pages 43–48, Jeju Island, Korea. Association for Computational Linguistics.

Begoña Villada Moirón and Jörg Tiedemann. 2006. http://aclweb.org/anthology//W/W06/W06-2405.pdf Identifying idiomatic expressions using automatic word-alignment. In *Proceedings of the EACL 2006 Workshop on Multiword Expressions in a Multilingual Context*, Trento, Italy.

Robert Östling and Jörg Tiedemann. 2016. http://ufal.mff.cuni.cz/pbml/106/art-ostling-tiedemann.pdf Efficient word alignment with markov chain monte carlo. *The Prague Bulletin of Mathematical Linguistics (PBML)*, (106):125—-146.