Opening OPUS for User Contributions

Jörg Tiedemann, Matthias Zumpe, Sebastian Schleussner

Department of Linguistics and Philology
Uppsala University
{firstname.lastname}@lingfil.uu.se

Abstract

OPUS is a growing collection of freely available parallel corpora. In this paper, we describe our on-going effort in developing a web application that opens the resource repository for external users who wish to contribute to the data collection. The system, we describe, implements a flexible, distributed repository backend that includes tools for automatic import of documents in a variety of formats and other features supporting data management and the creation of parallel aligned corpora. Currently, we work on a web frontend that implements the user interface to the repository software.

1. Introduction

OPUS (http://opus.lingfil.uu.se) is a widely used data collection of parallel corpora, tools and on-line corpus search interfaces. It currently covers over 90 languages and more than 40 billion tokens in 2.7 billion parallel translation units (Tiedemann, 2012). It is a growing resource mainly used by people working in data-driven machine translation (for example, statistical MT), cross-lingual corpus linguistics and translation studies. The demand of parallel corpora and related tools is high which can be seen in our web logs showing between 20,000 and 30,000 unique visitors every month. Our goal for the near future is to release a system that makes it easier to contribute to our collection. This system is designed to support all the major steps that need to be done when creating parallel aligned corpora from translated resources. The main purpose is to provide a simple interface to various tools that can be used to validate, convert and align documents contributed by external users in a variety of popular formats and to automatically create parallel data sets in the internal OPUS format. Such a system needs to be scalable and modular in order to cope with the requirements of a multi-user web application. Related initiatives collecting and integrating user-provided translation data exist but usually belong to commercial companies like Google (http://translate.google.com/toolkit) and MyMemory (http://mymemory.translated.net) or pay-services like Google (http://translate.google.com/toolkit) and MyMemory (http://mymemory.translated.net) or pay-services like Google (http://translate.google.com/toolkit) and MyMemory (http://mymemory.translated.net).

In the following, we briefly describe the repository backend, the data import features and the on-going work on a web frontend for OPUS.

2. The Resource Repository Backend

The resource repository is a highly modular toolbox that is designed to be flexible and scalable. Figure 1 illustrates the general architecture of the software. The communication with the web frontend is handled via HTTPS requests. The package provides several web service API’s to perform tasks at the backend and to obtain information from the resource collection. The backend system includes a central database server (DB server). Its task is to manage data permissions, group settings and the entire collection of meta information available for each resource in the repository. Each of the repository servers (RR servers) is connected to that database via a client-server architecture. For our purposes, we use a schema less key-value store, which makes it possible to easily extend and adjust the database without changing settings or internal data structures. The DB engine is based on TokyoCabinet (FAL, 2010), a modern implementation of a database management system that supports the storage of arbitrary data records without predefined typed data structures. Using the table extension of the software, we can attach records of key-value pairs to each resource identifier (which we use as unique table key). The query engine supported by TokyoCabinet is very powerful and fits our needs very well. Various search operators can be used to access key-value pairs and access is extremely fast. For remote access to the database server, we use TokyoTyrant, a network interface to TokyoCabinet. This server runs as a multi-threaded daemon on the DB server and TokyoTyrant clients connect from other RR servers via dedicated network ports.

Furthermore, RR servers are also configured as submit hosts of a general purpose grid engine. We use the Oracle Grid Engine (formerly Sun Grid Engine, SGE), a standard open-source implementation for Grid Computing, to perform tasks such as data conversion and alignment. In this
way, we can easily distribute jobs among several execution hosts, which makes the system highly scalable and extensible. For additional scalability, it is also possible to distribute resources over several storage servers. A load balancer at the frontend could be used to automate this task. However, such a component is not implemented yet.

Using our modular design, the system supports highly distributed setups. Resource repositories may be placed on local networks or even in a cloud-based solution. The database also supports replication on remote servers. RR servers and SGE execution hosts can be added on demand.

2.1 Data Upload

The resource repository supports the upload of documents in various formats. Each data upload triggers an automatic import process that includes a validation step, a conversion step and possibly another sentence alignment step in case parallel documents can be identified.

The software currently supports the following upload formats. Parallel data sets: Documents may already be aligned on some segmentation level (usually sentences). We support the three most common formats used in the translation business, the localization industry and in the machine translation community.

- **TMX**: Translation memories in translation memory exchange format (TMX). Character encoding is automatically identified and language specifications are taken from the language attributes of the translation units.
- **XLIFF**: Another translation memory format which is primarily used by the localization industry.
- **Moses format**: Archives of plain text files that are aligned on the sentence level. Languages are taken from the file extension as commonly used in the Moses SMT platform (Koehn et al., 2007).

TMX and XLIFF formats are checked with validating XML parsers. Minor problems are solved automatically using XML cleaning tools.

**Monolingual documents** can be in PDF format, various MS Word formats (doc and docx) and in plain text with various character encodings. Recently, we also added the support of the movie subtitle format SRT, HTML, generic XML and derived formats, the Open Document Format and RTF. The software uses standard tools such as Apache Tika (Mattmann and Zitting, 2011) for the conversion to text and, finally, our internal XML format. Character encodings are detected automatically using language-specific classifiers, byte-order markers and file content heuristics.

Each document added to a corpus in the repository is also checked by a language identifier (based on textcat [van Noord, 2010]). Mismatches with user specifications are marked in the metadata DB. The software supports also different settings of the automatic import. Import parameters can be controlled via user-specific, corpus-specific and resource-specific parameters. The same applies for the automatic sentence alignment. The system looks for parallel documents in the repository each time a new upload has been processed. In the default mode, only documents with identical names will be detected as parallel documents. However, the system also supports a fuzzy matching approach using name heuristics and file size ratios. The software also integrated several alignment tools (Gale and Church, 1993 Varga et al., 2005) and supports various settings of the underlying algorithms. Import and alignment processes can be restarted using different parameters using the Job API of the package.

3. The OPUS Web Interface

The task for the web interface is to provide an intuitive graphical system that allows external users to use the repository software and to inspect the data collections provided by its users. The interface is developed using a modern web framework called mojolicious (http://mojolicious.us/). This framework is a powerful toolbox supporting modern web technology including many plugins and extensions. The web interface supports all main features of the repository including data upload, download, inspection and other management tasks. It also implements a simple user management and is ready to be tested in a production environment.

4. Conclusions

In this paper we present the main components of a newly developed resource repository that will be used for the creation of parallel corpora from various sources. The system uses modern web technology with a distributed modular architecture in order to support external users when building new resources and contributing to OPUS.

5. References


