Towards a User-Friendly Webservice Architecture for Statistical Machine Translation in the PANACEA project*

Antonio Toral, Pavel Pecina, Andy Way
School of Computing
Dublin City University
Dublin, Ireland
{atoral,ppecina,away}@computing.dcu.ie

Marc Poch
IULA
Universitat Pompeu Fabra
Barcelona, Spain
marc.pochriera@upf.edu

Abstract

This paper presents a webservice architecture for Statistical Machine Translation aimed at non-technical users. A workflow editor allows a user to combine different webservices using a graphical user interface. In the current state of this project, the webservices have been implemented for a range of sentential and sub-sentential aligners. The advantage of a common interface and a common data format allows the user to build workflows exchanging different aligners.

1 Introduction

A human translator willing to use a state-of-the--art Statistical Machine Translation (SMT) system has two main options nowadays: to use an on-line translation system (e.g. Google Translate[1]) or to train his or her own MT system from scratch (e.g. Moses (Koehn et al., 2007)) using any available parallel corpus (e.g. Europarl[2]). Both options, however, present some important drawbacks. On one hand, the on-line systems are not customisable and might be inadequate if the user wants to translate sensitive data. On the other hand, installing an MT system with all its dependencies and training it requires technical expertise which is usually beyond the competences of an average translator.

This paper presents an architecture for building SMT systems where each component is deployed as a webservice. This way the user does not have to deal with technical issues regarding the tools, such as their installation, configuration or maintenance. A workflow editor allows the user to combine the different webservices using a graphical user interface. In the current state of this project, webservices have been created for a range of sentential and sub-sentential aligners.

This work is part of the FP7 PANACEA project,[3] which addresses the most critical aspect of MT: the language-resource bottleneck. Its objective is to build a factory of language resources that automates the stages involved in the acquisition, production, updating and maintenance of language resources required by MT systems. This is done by creating a platform, designed as a dedicated workflow manager, for the composition of a number of processes for LR production, based on combinations of different web services. The techniques developed are language independent, while the use case language-pairs are English–French, English–German and English–Greek.

The rest of the document is structured as follows. The following section details the implementation of webservices for each of the aligners considered. In addition, we discuss the procedures developed to convert the format of these aligners to a common format and present a set of workflows that demonstrate the usage of the aligners in processing pipelines. This is followed by a report on the software that has been developed. Finally, we draw conclusions from this experience and outline avenues of future work.

*We would like to thank the developers of Soaplab and Taverna for solving our questions and requests. This research has been partially funded by the EU project PANACEA (7FP-ITC-248064).

[1] http://translate.google.com/
[2] http://www.statmt.org/europarl/
[3] http://panacea-lr.eu

Mikel L. Forcada, Heidi Depraetere, Vincent Vandeghinste (eds.)
Proceedings of the 15th Conference of the European Association for Machine Translation, p. 63–70
Leuven, Belgium, May 2011

2 Aligners as Webservices

This section reports on the development of webservices for a range of widely-used state-of-the--art aligners which can be easily exchanged in user workflows due to the common interface designed within the project. Table 1 describes the mandatory parameters of the interface shared across all the webservices. In addition, a webservice might accept optional parameters that allow to exploit specific functionality of the aligner wrapped by that webservice.

Name	Type
source_language	2 character ISO code
source_corpus	text
source_language	2 character ISO code
target_corpus	text

Table 1: Shared mandatory parameters.

Webservices created for sentential aligners (Hunalign, GMA and BSA) are covered in section 2.1, while section 2.2 deals with sub-sentential aligners (GIZA++, BerkeleyAligner and OpenMaTrEx chunk aligner).

2.1 Sentential alignment

2.1.1 Hunalign

Hunalign (Varga et al., 2005)[4] can work in two modes. If a bilingual dictionary is available, this information is combined with sentence-length information (Gale and Church, 1991) and used to identify sentence alignment. In the absence of a bilingual dictionary, it first identifies the alignment using sentence-length information only, then builds an automatic dictionary based on this alignment and finally realigns the text using this dictionary.

This tool requires three parameters (filenames for the source corpus, target corpus and bilingual dictionary, although the dictionary file can be empty). As a webservice, the first two parameters are mandatory, together with the source and target languages. The bilingual dictionary file is optional, if none is provided the webservice will create and use an empty file. Hunalign also provides a set of optional parameters. Some of them are offered by the webservice (*bisent*, *cautious* and *text*), while two of them are activated internally (*realign* and *utf*). The remaining ones regard evaluation purposes or post-filtering and have been hidden from users of the webservice.

2.1.2 GMA

GMA – Geometric Mapping and Alignment (Argyle et al., 2004)[5] – is an implementation of the Smooth Injective Map Recognizer (Melamed, 1997) algorithm for mapping bitext correspondence and the Geometric Segment Alignment (Melamed, 1996) post-processor for converting general bitext maps to monotonic segment alignments. The tool employs word correspondences, cognates, as well as information from bilingual dictionaries.

This tool accepts a pair of parameters for the source and target corpus. Apart from that, it needs a parameter pointing to a configuration file, which contains several parameters, including language--dependent lists of stop words. The webservice offers two parameters for the source and target languages; these denote a language pair, which is internally assigned a configuration file. GMA provides configuration files and stop words for English–French. Additional configuration files have been created for the following language pairs: English–German, English–Spanish and English–Italian (all the parameters have the same values across language pairs except for those that are language-dependent). The stop word lists used for English, French, German, Spanish and Italian have been obtained from Université de Neuchâtel.[6]

2.1.3 BSA

BSA – Bilingual Sentence Aligner[7] (Moore, 2002) – is a three-step hybrid approach. First, sentence-length based alignment is performed; second, statistical word alignment model is trained on the high probability aligned sentences and third, all sentences are realigned based on the word alignments. It generates only 1:1 alignments.

This tool takes three parameters (filenames for the source corpus, target corpus and an alignment probability threshold). All of them are offered by the webservice developed. The threshold is set to *0.5* by default. The output of this tool was altered by modifying the script *filter-final-aligned-sents.pl*

[4] http://mokk.bme.hu/resources/hunalign

[5] http://nlp.cs.nyu.edu/GMA/

[6] http://members.unine.ch/jacques.savoy/clef/index.html

[7] http://research.microsoft.com/en-us/downloads/aafd5dcf-4dcc-49b2-8a22-f7055113e656/

which carries out the last phase of the alignment; it receives the set of word alignments found and outputs only those with alignment probability above the threshold. BSA originally outputs a couple of files with the aligned text (one sentence per line). Conversely, we are interested in a unique file where each line contains an alignment by giving the sentence numbers of the source and target languages (see section 3). The output format has been modified to that of Hunalign.

2.2 Sub-sentential alignment

2.2.1 GIZA++

GIZA++[8] (Och and Ney, 2003) is a statistical toolkit that is used to train IBM Models 1-5 and an HMM word alignment model. It performs word alignment in several steps that involve different tools (*plain2snt*, *mkcls*, *snt2cooc*, *GIZA++*, *giza2bal* and *symal*). The webservice developed encapsulates all of them through one call.

This word aligner toolkit offers many fine-grained input options, which are mainly numeric parameters that modify the behaviour of the aligner. Being the emphasis of the platform to provide easy-to-use versions of the tools, most of the parameters have been kept fixed (i.e. they cannot be changed through the webservice interface). These values (taken from Moses) are:

For *GIZA++*:
- *model1iterations 5*
- *model2iterations 0*
- *model3iterations 3*
- *model4iterations 3*
- *model1dumpfrequency 1*
- *model4smoothfactor 0.4*
- *nodumps 1*
- *nsmooth 4*
- *onlyaldumps 1*
- *p0* (parameter *p_0 in IBM-3/4*) *0.999*

For *symal*:
- *alignment grow*
- *diagonal yes*
- *final yes*
- *both yes*

Apart from the mandatory parameters that follow the common interface, the webservice interface offers two specific parameters for *mkcls*: number of iterations (default value *2*) and number of classes (default value *50*).

[8] http://code.google.com/p/giza-pp/

2.2.2 BerkeleyAligner

BerkeleyAligner[9] (Liang et al., 2006; DeNero and Klein, 2007; Haghighi et al., 2009) is a word alignment toolkit combining unsupervised as well as supervised approaches to word alignment. It features joint training of conditional alignment models (cross-EM), syntactic distortion model, as well as posterior decoding heuristics.

Similarly to GIZA++, this tool provides a plethora of options (see documentation/manual.txt in its distribution for details). Most of them are not considered in the webservice interface. The only parameters that are offered by the webservice are the mandatory ones following the common interface and the number of iterations to run the model (default value *2*).

2.2.3 OpenMaTrEx chunk aligner

OpenMaTrEx[10] is a marker-driven example-based machine translation system (Dandapat et al., 2010) that provides, among other capabilities, chunk alignment. The webservice developed performs chunk alignment by using several tools included in OpenMaTrEx, sequentially:

- Marker-based chunking (Veale and Way, 1997) (markers available for all the languages of the project but Greek are available).
- Word alignment, relying on GIZA++.
- Chunk alignment, using an algorithm based on Levenshtein edit distance (Levenshtein, 1966) and employing cognates and word probabilities as distance knowledge.

The parameters offered by the webservice are the mandatory ones according to the common interface. The output is produced by running the tool in the *ebmt_alignments_on_disk_with_id mode*; this mode carries out chunk alignment adding sentence identifiers to the output and has been modified to provide not only the identifier of the sentence but also the identifiers of the tokens that delimit each chunk in each language (OpenMaTrEx outputs the textual chunks instead of their numeric identifiers).

3 Travelling Object

This section deals with the conversion of the different input/output formats of the aligners from and

[9] http://code.google.com/p/berkeleyaligner/

[10] http://openmatrex.org/

to a common format, the Travelling Object (TO) (Poch et al., 2010), which is based on the schema for alignment in XCES.[11]

A sample output for word alignment is shown in Figure 1. It aligns two documents (*doc_en.xml* and *doc_es.xml*). There are two groups of links for the first sentence of each file. The first token in the source document is aligned to the first in the target. The range of tokens two to three in the source are aligned to the second token in the target. The same format can be used for sentential alignment changing the values of the domains from sentences to paragraphs and the targets from tokens to sentences.

Several webservices have been developed to deal with the conversion between the formats of the aligners and the TO. *aligner2to* is a webservice that converts the output of any of the aligners to the TO. Another webservice, *sentalg_tok_to2word_alg*, takes three inputs in TO format (source and target corpus, both sentence--split and tokenised, and sentence alignment) and outputs two files (the subset of sentences in the source and target corpus that have 1-to--1 sentence alignments) in the plain sentence--split and tokenised format that the sub-sentential alignment webservices take as input. Finally, *sentsplit_tok2to* is a webservice that was required by *sentalg_tok_to2word_alg*; it converts sentence--split and tokenised files to the TO format.

4 Workflows

This section presents several workflows of webservices showing possible usages of aligners in the platform, the interactions that arise, etc. These workflows have been created using Taverna[12] (Hull et al., 2006), a Workflow Management System that allows the user to create workflows by combining webservices using a graphical interface.

The first workflow, depicted in Figure 2, presents a pipeline that performs sentence alignment on an English–Spanish parallel corpus (inputs *url_sl* and *url_tl*). Each side of the corpus is preprocessed with a sentence splitter (*europarl_sentence_splitter*) and a tokeniser (*europarl_tokeniser*). Then the corpus is sentence-aligned using Hunalign. Finally, the outputs of the alignment and the tokenisers are converted to the TO using *aligner2to* and *sentsplit_tok2to* respectively.

Figure 3 and Figure 4 show modified versions of the workflow presented in Figure 2 to perform sentence alignment with the other sentence aligners, BSA and GMA, respectively. Figure 5 shows a workflow that performs word alignment using GIZA++. The workflow in Figure 6 carries out word alignment using BerkeleyAligner. Figure 7 performs chunk alignment using OpenMaTrEx.

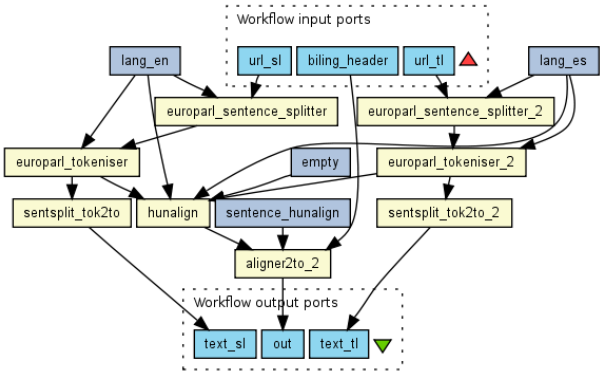

Figure 2: Sentence alignment using Hunalign.

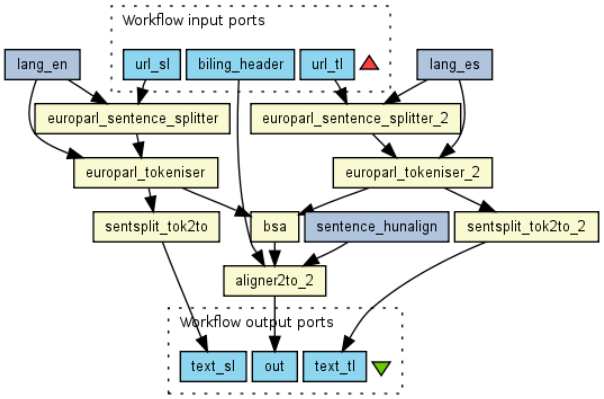

Figure 3: Sentence alignment using BSA.

[11] http://www.xces.org/schema/#align
[12] http://www.taverna.org.uk/

```
<cesAlign version="1.0"  xmlns="http://www.xces.org/schema/2003">
  <cesHeader version="1.0">
    <profileDesc>
      <translations>
         <translation trans.loc="doc_en.xml" wsd="UTF-8" n="1"/>
         <translation trans.loc="doc_es.xml" wsd="UTF-8" n="2"/>
      </translations>
    </profileDesc>
  </cesHeader>
  <linkList>
    <linkGrp domains="s1 s1" targType="t">
      <link>
        <align xlink:href="#s1_t1"/>
        <align xlink:href="#s1_t1"/>
      </link>
      <link>
        <align xlink:href="#xpointer(id('s1_t2')/range-to(id('s1_t3')))"/>
        <align xlink:href="#s1_t2"/>
      </link>
      [...]
    </linkGrp>
  <linkList>
</cesAlign>
```

Figure 1: Travelling Object sample for word alignment.

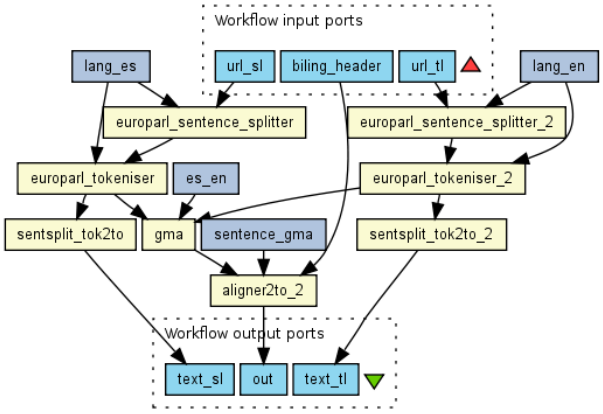

Figure 4: Sentence alignment using GMA.

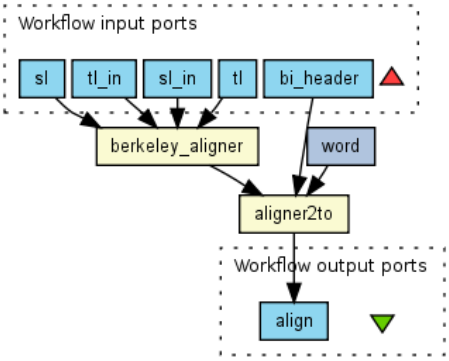

Figure 6: Word alignment using BerkeleyAligner.

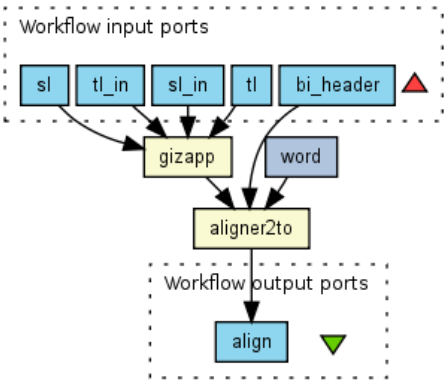

Figure 5: Word alignment using GIZA++.

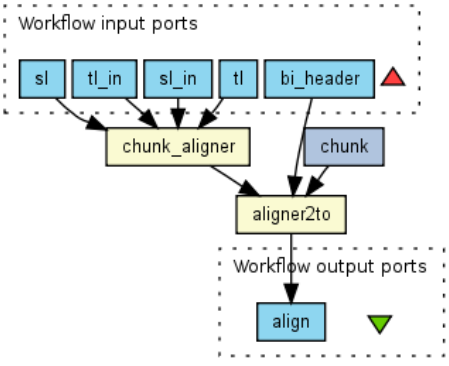

Figure 7: Chunk alignment using OpenMaTrEx.

Finally, Figure 8 presents a combined pipeline that performs both sentence and word alignment. It consists of a pipeline made up of three nested workflows that execute sequentially: sentential alignment, conversion from sentence alignment TO-compliant output to plain-text word alignment input, and word alignment. The webservices used for alignment are Hunalign for sentential alignment and GIZA++ for word alignment. However, thanks to the use of the common interface and the TO format, any of the aligners could be easily used to replace these.

5 Software

The software and data developed can be classified in two categories: webservices and workflows.

The webservices are developed using Soaplab2.[13] This tool allows to deploy webservices on top of command-line applications by writing files that describe the parameters of these services in ACD format.[14] Soaplab2 then converts the ACD files to XML metadata files which contain all the necessary information to provide the services (script to be run, parameters, options, help messages, etc.). The Soaplab server is a web application run by a server container (Apache Tomcat[15] in our setup) which is in charge of providing the services using the generated metadata. The software pipeline for each webservice reflects this schema:

$$ACD\_file \rightarrow script \rightarrow tool\_binary$$

That is, an ACD file is created to define the input/output ports of the soaplab2 webservice. This file links to an intermediate script, which, regardless of the webservice, is in charge of three main tasks:

- Parameter handling. The parameters offered by the corresponding webservice are checked; if any of them are missing or if any different parameter is used, the execution is aborted.
- Security. Execution is automatically aborted if any command fails or if any variable is not set.
- Logging. A folder is created for each run and holds the input and output files and any log produced by the tool itself.

Finally the tool that the webservice wraps is called from the script. Soaplab is used in many fields like bioinformatics because of its ease of use, low cost maintenance and features. With Soaplab, web service providers do not need to be expert programmers, as they only need to describe their tools with metadata.

All the software developed is released under the GPL version 3[16] and can be accessed at `http://www.computing.dcu.ie/~atoral/#Resources`. The source files of the workflows detailed in section 4 are also included.

6 Conclusions and Future Work

This paper has presented a webservice architecture for Statistical Machine Translation aimed at non-technical users. Compared to the existing on-line systems, the proposal allows users to customise systems to their own personal needs. Compared to the available decoders, it has the advantage of lowering the level of expertise required to build an MT system, thus allowing less computer-literate users to benefit from access to the state-of-the-art SMT systems used by researchers and system builders today. Moreover, the hardware and software maintenance cost for the user is dramatically lowered since most of the software is executed on remote machines where resources (cpu, memory, etc.), temporary files and software updates are being managed.

In its current status, the architecture supports alignment, both sentential and sub-sentential. Webservices have been developed for a range of state-of-the-art aligners. Each webservice acts as a wrapper over an aligner, and offers a subset of its original functionality (the aim being to provide easy-to-use tools for final users).

Sample workflows for each of the aligners have been presented. These give a glimpse of the potential of the architecture for the final user. They also show how the aligners interact with the other webservices.

An important role is devoted to interoperability. The usage of a common interface for all the aligners and of a common input/output format allows the user to use any of the provided aligners.

[13] `http://soaplab.sourceforge.net/soaplab2/`
[14] `http://soaplab.sourceforge.net/soaplab2/MetadataGuide.html`
[15] `http://tomcat.apache.org/`
[16] `http://www.gnu.org/licenses/gpl-3.0.html`

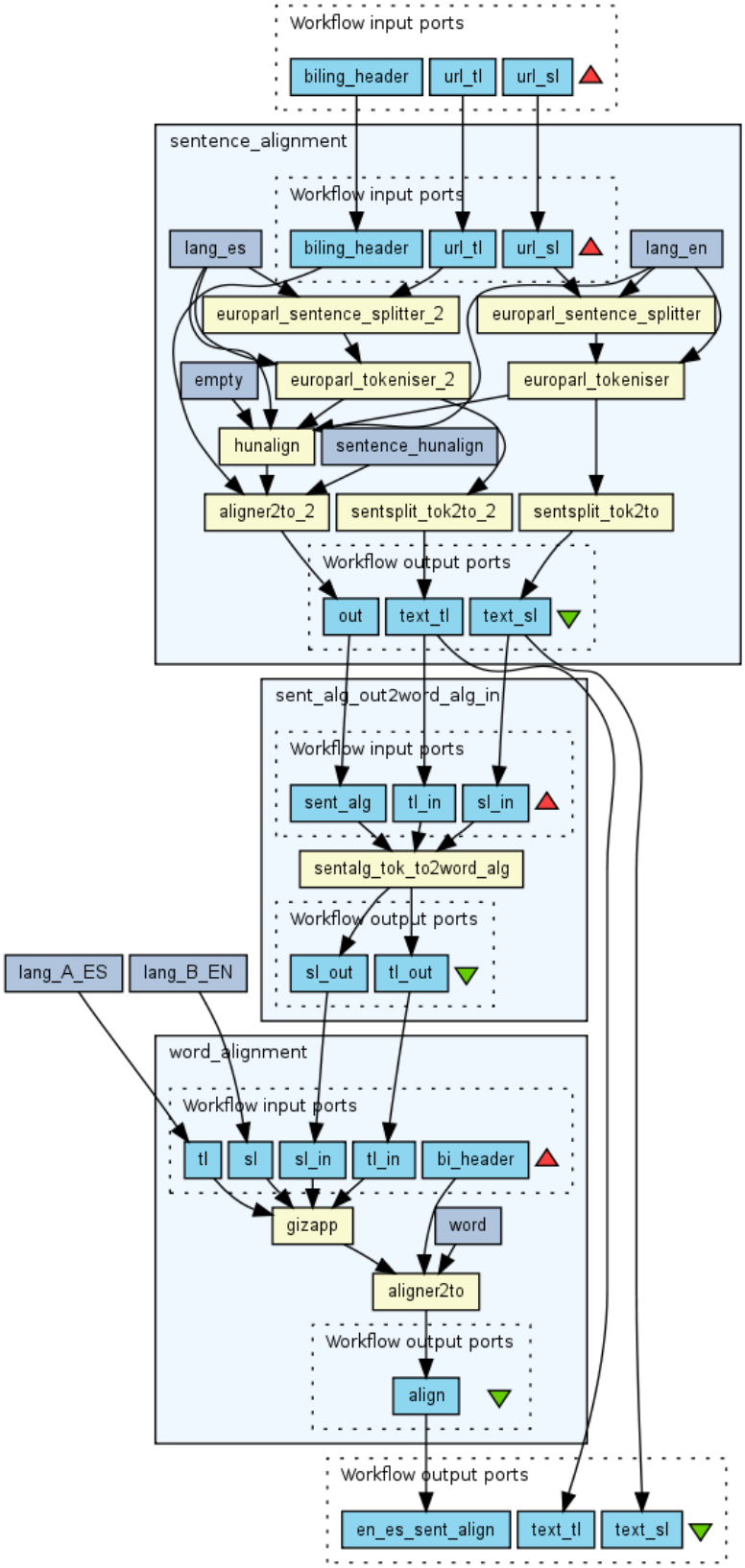

Figure 8: Pipeline that performs sentence and word alignment.

Regarding future work, we consider three main lines: First, a user validation to measure to what extent the architecture meets the expectations of users is currently under way. Second, we plan to incorporate components for the remaining steps of a SMT pipeline, mainly training and decoding. Finally, a real-world usage of this architecture will require the establishment of limitations on the server side, which might be system-wide (e.g. taking into consideration load, memory, processing time or number of concurrent users) or per webservice (e.g. taking into account the size of the input). This kind of architectures open the door to new exploitation models for tool developers or service providers who could be responsible for maintaining those tools while helping the research community.

References

Dandapat, Sandipan, Mikel L. Forcada, Declan Groves, Sergio Penkale, John Tinsley, and Andy Way. 2010. Openmatrex: a free/open-source marker-driven example-based machine translation system. In *Proceedings of the 7th international conference on Advances in natural language processing*, IceTAL'10, pages 121–126, Berlin, Heidelberg. Springer-Verlag.

DeNero, John and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Gale, William A. and Kenneth W. Church. 1991. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, ACL '91, pages 177–184, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 923–931, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hull, Duncan, Katherine Wolstencroft, Robert Stevens, Carole Goble, Matthew Pocock, Peter Li, and Thomas Oinn. 2006. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web Server issue):729–732, July.

Koehn, Philipp, 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. 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.

Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

Melamed, I. Dan. 1996. A geometric approach to mapping bitext correspondence. In *Conference on Empirical Methods in Natural Language Processing*.

Melamed, I. Dan. 1997. A word-to-word model of translational equivalence. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, EACL '97, pages 490–497, Stroudsburg, PA, USA. Association for Computational Linguistics.

Moore, Robert C. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, AMTA '02, pages 135–144, London, UK. Springer-Verlag.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29:19–51, March.

Poch, Marc, Prokopis Prokopidis, Gregor Thurmair, Carsten Schnober, Riccardo Del Gratta, and Núria Bel. 2010. D3.1 - architecture and design of the platform. Confidential deliverable, The PANACEA Project (7FP-ITC-248064).

Varga, Dániel, László Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. 2005. Parallel corpora for medium density languages. In *Proceedings of the Recent Advances in Natural Language Processing*, pages 590–596, Borovets, Bulgaria.

Veale, Tony and Andy Way. 1997. Gaijin: A Bootstrapping, Template-Driven Approach to Example-Based MT. In *Proceedings of New Methods in Natural Language Processing*, pages 239–244, Sofia, Bulgaria.