Using the Stockholm TreeAligner

Joakim Lundborg, Torsten Marek, Maël Mettler*, Martin Volk
Stockholm University
Department of Linguistics

Abstract

In this paper we present several use cases for the Stockholm TreeAligner, a software tool originally designed for annotating the alignments in a parallel treebank. The tool has been extended and improved to the point that it can now also serve as a general tool for browsing and searching monolingual and parallel treebanks.

Among the use cases presented are: building a parallel treebank, browsing mono- and bilingual treebanks, consistency checking using the search function, comparing PP-attachment in different languages, and viewing different versions of the same treebank. A demonstration of the software will be held during the workshop.

1 Introduction

There is a rising interest in building parallel corpora, that is corpora consisting of texts translated to several languages. Such resources are valuable for language technology, linguistics and translation science. But as of yet there are only few parallel corpora available, in few languages, covering few text genres.

Parallel corpora can be of many kinds, in some cases only the words are parallel, i.e. words that are translations of each other are annotated accordingly. Others annotate alignments between syntactical sub-sentential elements, for example using phrase structure trees.

The software tool described in this paper, the Stockholm TreeAligner, was developed for and while creating a syntactically annotated parallel corpus; the SMULTRON treebank. In this treebank, sub-sentential elements (non-terminals) are aligned in addition to just the words (terminals).

The TreeAligner has recently gained querying support, including searching for alignments (see (Volk, Lundborg, and Mettler, 2007) for a description of the query functionality). Searches are evaluated using a subset of the query language in TIGERSearch (Wolfgang Lezius, 2002), but with additions for searching alignments. The TIGERSearch query language[1] has proven its use and provides us with a solid foundation for a tree query language as well as inspiration for the implementation.

*This paper is a tribute to Maël Mettler who died unexpectedly in October 2007. His contributions to the paper and to the software (in particular the query module) but even more his enthusiasm for our project will always be remembered.

[1]See `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/`

De Smedt, K., Hajič, J. and Kübler, S. (Eds.)
Proceedings of the Sixth International Workshop
on Treebanks and Linguistic Theories (2007)

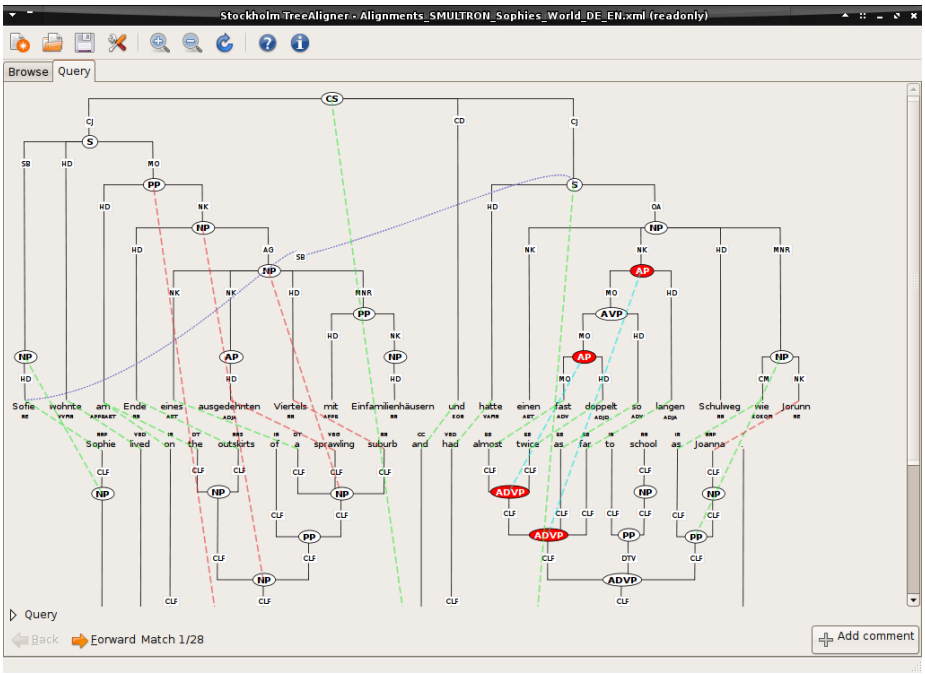

Figure 1: A tree pair annotated using Stockholm TreeAligner. Green lines represent exact translation correspondence, and red represent approximate correspondence. Results of a search query is highlighted with red node labels and tinted alignment lines.

TIGERSearch was implemented in Java. In contrast, the Stockholm TreeAligner is implemented in Python, which both increases developer productivity and results in programs of considerably less size. As an indication of the difference, the TIGERSearch source distribution contains 69 100 lines of code while the TreeAligner only contains 4 989.[2]

With the addition of alignment searches, the TreeAligner is a powerful tool for creating, browsing and searching parallel treebanks, useful in several cases when working with parallel treebanks. We will here describe such cases.

2 Building a parallel treebank

The primary design goal for the Stockholm TreeAligner was creating an easy-to-use environment to manually annotate the alignments in a parallel treebank. Many of the features available come as a direct consequence of this design goal. The input to the TreeAligner consists typically of two treebanks that are built beforehand. The treebanks need to be in the TIGER-XML format used by TIGERSearch. If the two treebanks have been automatically aligned on some level, this information can also be imported into the TreeAligner.

[2]This comparison was done using the Sloccount utility on TIGERSearch version 2.1.1 and the latest development version of the Stockholm TreeAligner. Sloccount only counts actual lines of code, not whitespace or comments, and it is freely available at `http://www.dwheeler.com/sloccount/`

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace/handle/10062/4476

2.1 Annotating

Annotating the alignments in the software is easy; once you have the trees that contain nodes to be aligned (i.e. syntactical elements or words), you simply drag a line with your mouse between the elements you wish to annotate as having an alignment relation. In this way, it is possible to create one-to-one, as well as one-to-many relations.

We allow different types of alignment annotation, in the current stage "good" and "fuzzy" alignments. In the future, the program shall support additional types of alignments, to indicate different kinds of relations between the nodes. An example of an annotated tree as displayed by Stockholm TreeAligner is in Figure 1. The alignment information is stored in an XML file, with a flexible and simple format. The TreeAligner can also export the trees into SVG, PDF and PNG formats, enabling it to produce graphics for use in publications etc.

2.2 Commenting

The TreeAligner allows the user to attach a comment to any tree pair. Comments enable us to document the building process by marking difficult cases, sentences with similar constructions and other notable cases.

Comments are also helpful in the finished treebank, to identify a particularly interesting tree, indicating semantic ambiguity as well as other information that could be helpful for the user of the treebank.

End-users of the treebank are also able to use this feature, for example to mark interesting trees for themselves.

2.3 Related Work

The Stockholm TreeAligner can be seen as an advancement over tools like Cairo (Smith and Jahr, 2000) or I*Link (Merkel, Petterstedt, and Ahrenberg, 2003) which were developed for creating and visualising word alignments but were unable to display trees. (Samuelsson and Volk, 2007) gives an overview of such tools. The TreeAligner is even more related to Yawat and Kwipc (Germann, 2007), two software tools for the creation and display of sub-sentential alignments in matrices. Our tool is unique in that it displays the complete tree structures and allows powerful queries over the trees and the alignment.

3 Querying

The TreeAligner query system works by using three different inputs; one query for each of the treebanks, and one for the alignment relation between these queries. We use the TIGERSearch syntax for referring to the different parts of the query like this:

```
Treebank1   #node1: [cat="NP"]
Treebank2   #node2: [cat="NP"]
Alignment   #node1 * #node2
```

Here #node1 is a variable that identifies a node of category NP in a tree in treebank1. And #node2 identifies a node of category NP in treebank2. These

variables correspond exactly to the syntax in the TIGERSearch query language. We then use these variables in the alignment query. The general alignment relation is indicated by the "*" operator. For a detailed description of the alignment query syntax, see (Volk, Lundborg, and Mettler, 2007). (Wolfgang Lezius, 2002) describes the TIGERSearch query language on which we based our query language (note that some parts of the TIGERSearch query language, e.g. negation, are not yet implemented in the TreeAligner).

3.1 Using the search function for consistency checking

For a parallel treebank to be useful we need to make sure that the alignment annotation is done consistently across all of the tree pairs.

Doing consistency checking by manually browsing all alignments that involve a particular construction is not practical. The search function can be used to do a rudimentary form of consistency checking of alignments.[3]

Let us look an example. While building our treebank we found that in some cases we had aligned English VP nodes to Swedish S nodes. This is not correct in general since a VP does not contain a subject. To find out if we had made such annotations we constructed the following query:

```
Treebank1   #node1: [cat="VP"]
Treebank2   #node2: [cat="S"]
Alignment   #node1 * #node2
```

This query says, find all VPs in Treebank1 (English in our case) that are aligned to an S node in Treebank2 (Swedish). Using this query, finding and fixing all the erroneously annotated alignments is easy. Other kinds of consistency checks can also be carried out by writing appropriate queries.

3.2 Automation

It is desirable to automate the annotation process as much as possible. This will become easier once we have manually annotated some data which will then allow us to use data-driven annotation methods. Such efforts are already on the way in our research group. Automated word and phrase alignment is planned to be integrated within the TreeAligner to help the annotator, yielding a semi-automatic alignment process. The TreeAligner can also be used to browsing and correcting automatically aligned treebanks.

4 Browsing and searching monolingual treebanks

When constructing the interface for annotating the alignments, it was also necessary to make the TreeAligner a good browser for treebanks, so that the annotator is able to see the trees, including all syntactic information. The support for browsing and searching monolingual treebanks is a trivial side effect.

[3]More advanced consistency checking could be done by combining this querying with established statistical consistency checking methodology, but we have not yet done so.

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476

5 Browsing dependency treebanks

The TreeAligner was originally designed for the alignment annotation in constituent structure treebanks, but we have recently also added support for working with dependency treebanks. Dependency trees can be drawn in several ways. We chose to start by drawing them in a fashion similar to constituent trees, since that allowed us to re-use our tree drawer. We may extend this in the future to support drawing of dependency trees with the curved arrow style instead.

Doing the actual alignment in such treebanks is less complex, since there are no sub-sentential syntactic elements to worry about; all alignments are word alignments. There is one large parallel dependency treebank available (see (Martin Cmejrek, Jan Curín, and Jirí Havelka, 2005)). The TreeAligner can be used to browse and search such parallel treebanks graphically, given that they can be converted to the appropriate input format.

6 Comparing attachment decisions across languages

One of the purposes of having a parallel treebank is to be able to do cross-language studies of syntactic structures, for example how the same meaning unit is expressed syntactically in different languages. This is a short example of such an investigation.

In this example, we are interested in cases where a noun-modifying PP in one language is aligned to an adjective phrase in the other language. This can be accomplished by a search query like:

```
Treebank1   #np1:[cat="NP"] > #ap:[cat="AP"]
Treebank2   #np2:[cat="NP"] > #pp:[cat="PP"]
Alignment   #np1 * #np2   &   #ap * #pp
```

This query will match AP nodes dominated by an NP in language 1 that are aligned to PP nodes in language 2 dominated by an NP that is also aligned to the NP matched in language 1. This will find cases where the modifier of an NP is expressed as an AP in one language and as a PP in the other language.

With good search queries, investigations like these are easy to do in parallel treebanks.

7 Viewing differences between treebanks

At one point in the development of our parallel treebank, the alignment process made apparent some errors in the initial syntactical annotation. We had to change the treebanks to be able to do consistent alignment. But this made the alignments already created erroneous in some cases.

We found that loading two versions of the same treebank into TreeAligner gave a good overview of the changes (they were easy to spot with the trees next to each other). We prepared a small Perl script to visualise the nodes that were differently annotated. This also gave the opportunity to annotate and comment the changes in the trees.

This functionality is so useful that we plan to make this an integral part of the software in the future (using a more advanced tree comparison algorithm).

8 Conclusion

The Stockholm TreeAligner has proven to be useful way beyond the initial goals of its development; constructing the SMULTRON parallel treebank[4]. It now has the potential of replacing a number of tools that we have previously used in our work flow (e.g. TIGERSearch which only works on monolingual treebanks).

The software is developed in an open process, and is distributed under the free software license GPL [5]. You are invited to help the further development by sending your suggestions on improvements, bug reports, use cases, or by simply joining the development team.

References

Germann, Ulrich. 2007. Two tools for creating and visualizing sub-sentential alignments of parallel text. In *Proc. of The Linguistic Annotation Workshop at ACL 2007*, pages 121–124, Prague.

Martin Cmejrek, Jan Curín, and Jirí Havelka. 2005. Prague Czech-English dependency treebank. Resource for structure-based MT. In *Proceedings of EAMT 10th Annual Conference, Budapest.*

Merkel, Magnus, Michael Petterstedt, and Lars Ahrenberg. 2003. Interactive word alignment for corpus linguistics. In *Proc. of Corpus Linguistics 2003*, Lancaster.

Samuelsson, Yvonne and Martin Volk. 2007. Alignment tools for parallel treebanks. In *Proceedings of GLDV Frühjahrstagung 2007.*

Smith, Noah A. and Michael E. Jahr. 2000. Cairo: An alignment visualization tool. In *Proc. of LREC-2000*, Athens.

Volk, Martin, Joakim Lundborg, and Maël Mettler. 2007. A search tool for parallel treebanks. In *Proc. of The Linguistic Annotation Workshop (LAW) at ACL*, Prague, June.

Wolfgang Lezius. 2002. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora.* Ph.D. thesis, University of Stuttgart.

[4]The SMULTRON treebank is now available from `http://www.ling.su.se/dali/`

[5]See `http://www.gnu.org/licenses/gpl.html`

ISSN 1736-6305 Vol. 1
http://dspace.utlib.ee/dspace
/handle/10062/4476