Combining Diverse Word-Alignment Symmetrizations Improves Dependency Tree Projection

David Mareček

Charles University in Prague,
Institute of Formal and Applied Linguistics
marecek@ufal.mff.cuni.cz

Abstract. For many languages, we are not able to train any supervised parser, because there are no manually annotated data available. This problem can be solved by using a parallel corpus with English, parsing the English side, projecting the dependencies through word-alignment connections, and training a parser on the projected trees. In this paper, we introduce a simple algorithm using a combination of various word-alignment symmetrizations. We prove that our method outperforms previous work, even though it uses McDonald's maximum-spanning-tree parser as it is, without any "unsupervised" modifications.

1 Introduction

Syntactic parsing is one of the basic tasks in natural language processing. The best parsers learn grammar from manually annotated treebanks and for all there holds the rule that increasing amount of training data improves their performance. However, there are many languages for which a very few linguistic resources exist. Developing a new treebank is quite expensive and for some rarer languages it is even impossible to find linguists for the annotations.

In recent years, numerous works have been devoted to developing parsers that would not need much annotated data. One way is the totally unsupervised parsing (e.g. the Klein and Manning's inside-outside method [1]), which infers the dependencies from raw texts only. But the performance of such parsers is quite low so far. This changes when we append a few annotated sentences to the raw texts. Koo proved in [2] that this causes a great improvement.

Hwa came up with an idea [3] to use a parallel corpus. For many languages there exist some form of parallel texts, very often with English or other resource-rich languages being the coupled. The idea is to make a word-alignment, parse the English side of the corpus, then project the dependencies from English to the other language using the alignment, and, finally, train a parser on the resulting trees or tree fragments. Several similar works ([4], [5], [6], [7]) came after and made many improvements on this process. Ganchev [6] and Smith and Eisner [5] combine this method with unsupervised training.

A. Gelbukh (Ed.): CICLing 2011, Part I, LNCS 6608, pp. 144–154, 2011.

In all our experiments, we project dependency trees from English to other languages, denoted by the attribute *foreign* or letter X throughout the text. However, our approach should be effortlessly applicable also for language pairs with other source language.

When implementing the word alignment task in a parallel corpus, one can hardly expect only perfect 1:1 alignment links (e.g. due to typological language differences). If M:N links are allowed, a wide scale of alignment link types arises. Some of them are necessary or reasonable from one viewpoint, but spurious from the other (e.g. when aligning functional words). Hence, the fact that it is difficult to get one single ultimate word alignment, is not implied only by technical imperfectness of current implementations, but rather by the nature of languages. Simply said, different alignment schemes must be for different tasks. However, this paper shows that we can profit from the diversity.

The novel contribution of this paper lies in exploiting several types of word-alignment links; the previously published works expected a single word alignment on the input, but we show that combining more asymmetric sentence alignments leads to better results. Different types of alignment links can imply different reliability of projected edges, which provides the projection procedure with additional information. Furthermore, we also establish a method for filtering out the noise before training the parser. We use so called *alignment sparseness* and *non-projectivity* metrics for filtering sentences.

In Section 2, we comment previous works related to dependency projection. Section 3 describes word-alignment symmetrizations and discuss their suitability. Our projection algorithm is described in Section 4 and the process of data filtering is in Section 5. In Section 6, we present parsing accuracies on various languages and compare them to previous works. We conclude in Section 7.

2 Related Work

Our projection method was inspired by Hwa's work [4] from 2005, in which word alignment was used for projecting dependencies from English into Spanish and Chinese. They solved the "more counterpart" problem (what to do if an English word has more than one corresponding word) by choosing the left-most corresponding word as a main node; each other corresponding word then becomes dependent on the previous one (left-to-right dependencies). Unlike us, they used only one type of alignment link and they did not specify which method of symmetrization they used. They also introduced the data pruning criteria for filtering out the sentences where the alignment is bad (e.g. too many not aligned words or too many counterparts for one word) and added some hand-written rules to handle heterogeneity of different annotation schemas.

Smith and Eisner introduced in [5] quasi-synchronous grammar features for dependency projection and adaptation of annotation.

Ganchev et al. use in [6] so called "conserved dependencies", in which counterparts of governing and dependent word form a dependency edge (with the same orientation) in English. They ran an unsupervised parser with posterior regularization, in which inferred dependencies should correspond to projected ones.

Jiang and Liu use in [7] a matrix with alignment probabilities instead of the alignment itself in their projection algorithm. They compute a score for each possible link between the two parallel sentences and then, using a threshold, train a parser on the projected dependencies. They made the evaluation on the same Chinese data as in [4] and obtained better accuracy.

3 Alignment Symmetrization Methods

We use GIZA++ tool [8] to make word-alignments in parallel corpora. The alignment created by GIZA++ is asymmetric. For each word in one language just one counterpart in the other language is found, as it is depicted in Figure 1. Standard practice in machine translation tools is to run this alignment twice in both the directions (source-to-target and target-to-source) and use one of symmetrization methods described in [8]. For example, if we make an intersection of the two asymmetric alignments in Figures 1 and 2, we get the symmetric alignment, which is in Figure 3.

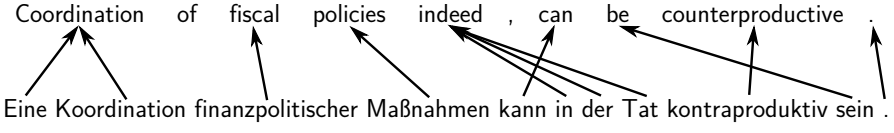

Fig. 1. German-to-English alignment example. From each word in the German sentence "Eine *(A)* Koordination *(coordination)* finanzpolitischer *(fiscal)* Maßnahmen *(policies)* kann *(can)* in *(in)* der *(the)* Tat *(fact)* kontraproduktiv *(contraproductive)* sein *(be).*" a link is made to just one English word.

Our task is different from machine translation. We do not need to make any symmetrization, because the task itself is asymmetric. We have a parse tree in English and we want to project it to the other (foreign) language. We would like to know a counterpart for each foreign word, because each foreign word must depend on some other foreign word in the new tree. On the other hand, we do not need to know a counterpart for each English word. From this point of view, the asymmetric alignment *X-to-English* (Figure 2) seems to be more useful for the projection of dependencies than the opposite alignment *English-to-X*.

Of course, symmetrized alignment is useful too. If a connection between two words appears in both *X-to-English* ($XtoEN$) and *English-to-X* ($ENtoX$), it should be more preferred. Beside the *intersection* (INT) alignment, which is

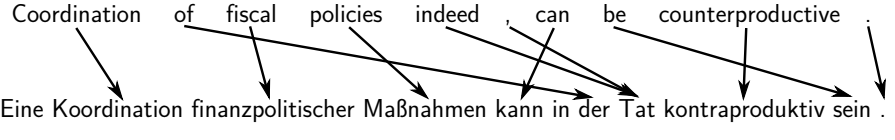

Fig. 2. English-to-German alignment example. A link is made from each English word.

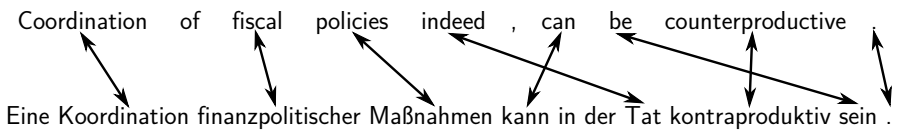

Fig. 3. Example of intersection alignment

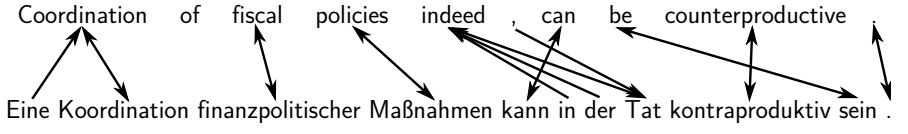

Fig. 4. Example of grow-diag-final-and alignment

depicted in Figure 3, we will use in this work also so called *grow-diag-final-and* ($GDFA$) alignment (Figure 4), in which there are all links from the intersection alignment and some other links adjacent to already added links. All the symmetrization methods are described in [8].

4 Algorithm for Projecting Dependency Trees

In this section, we describe our projection algorithm in detail. We present the setting which led to the best projection results across languages we tested. However, it is possible that a slightly different setting would be more useful for some other languages.

4.1 Assignment of Corresponding Words

First of all, we go through the English sentence and for each English word e_i we find a set of corresponding foreign words $C(e_i) = [f_{c_1}, f_{c_2}, \ldots]$. The set can be empty as well as it can contain more than one foreign word. But every foreign word can belong at most to one English word.

The foreign words f_{c_j} in the sets $C(e_i)$ are ordered according to the type of alignment connection between f_{c_j} and e_i. In the first position, there is a word connected by an *intersection* link (if it exists). This links have the highest weight since they are confirmed by both GIZA++ runs. They are followed by words connected by alignment links *X-to-English* that are also in *grow-diag-final-and* alignment. Words connected only by *X-to-English* links are at the end. The whole procedure is described in detail in Figure 5. The first three loops add words into sets sequentially; the last loop searches for English words that have no correspondent so far and if such English word e_i is linked to some word f_j and f_j is not the only word in its set, f_j is transfered to $C(e_i)$.

This assignment method ensures that each foreign word now belongs to just one English word. In this point we differ from previous works. This helps in searching dependencies for words that do not have its own counterpart in English (mostly the function words). Such word then becomes dependent on a word with

Input: INT, $GDFA$, $XtoEN$, $ENtoX$ ... various alignments of a sentence
Output: C ... sets of correspondent words for each English word

foreach $[e_i, f_j] \in INT$ **do**
    Push f_j to $C(e_i)$;
end
foreach $[e_i, f_j] \in (GDFA \cap XtoEN) \setminus INT$ **do**
    Push f_j to $C(e_i)$;
end
foreach $[e_i, f_j] \in XtoEN \setminus GDFA$ **do**
    Push f_j to $C(e_i)$;
end
foreach $[e_i, f_j] \in (ENtoX \cup GDFA) \setminus INT$ **do**
    $e_k \leftarrow$ such English node for which $f_j \in C(e_k)$;
    if $C(e_k)[0] \neq f_j$ **then**
        Delete f_j from $C(e_k)$;
        Push f_j to $C(e_i)$;
    end
end

Fig. 5. Algorithm for assignment of corresponding words

which it shares its counterpart. Since the shared counterpart is only one (it is determined by the *X-to-English* alignment), we do not need to use any heuristic for choosing one as it is in [4].

To conclude, the acquired sets of corresponding words are very close to the *X-to-English* alignment, only some connections are substituted by *English-to-X* links so that more English words would be covered. Of course, there are many other possibilities how to deal with different alignment symmetrizations, but this method seems to be the best for our testing languages.

4.2 Building the Dependency Tree

The algorithm for building the dependency tree of a foreign sentence consists of one recursive function `project_subtree()`. It goes through the English tree in a depth-first manner and builds the foreign tree at the same time. The process is described in pseudo-code in Figure 6. The example of an English-to-German projection is depicted in Figure 7.

When an English node e_i is processed, we choose from the ordered set of corresponding words $C(e_i)$ the first one and declare it as the *main counterpart*. The other corresponding words then become its children.

We demonstrate the algorithm on an example in Figure 7. The English node *indeed* has three corresponding words in the German sentence: *in*, *der*, and *Tat*. While the word *Tat* is connected to *indeed* by an *intersection* link, the other two words are connected only with *grow-diag-final-and* links. This means that $C(indeed) = [Tat, in, der]$ and *Tat* becomes the main counterpart of *indeed* and the words *in* and *der* become its children.

```
e_root = technical root of English parse tree;
f_root = technical root of foreign parse tree;
build_subtree(e_root, f_root);

function build_subtree(e_node, f_node);
begin
    foreach e_child ∈ children(e_node) do
        if |C(e_child)| = 0 then
            build_subtree(e_child, f_node);
        else
            main_f_child ← C(e_child)[0];
            parent(main_f_child) ← f_node;
            foreach f_child ∈ C(e_child) do
                if f_child ≠ main_f_child then
                    parent(f_child) ← main_f_child;
                end
            end
            build_subtree(e_child, main_f_child);
        end
    end
end
```

Fig. 6. Algorithm for projection of dependency trees

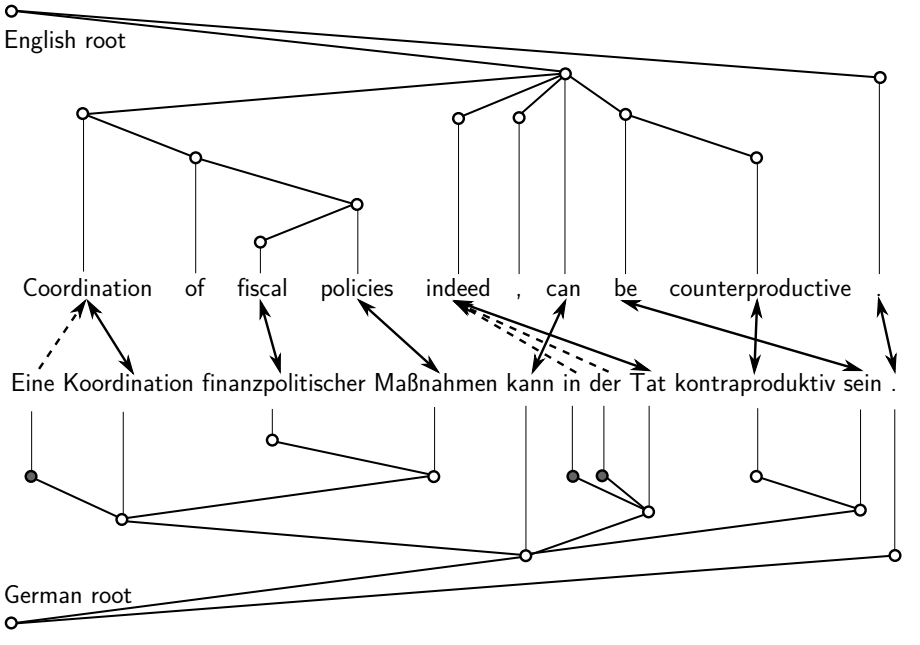

Fig. 7. Projection of an English dependency tree into German. Intersection connections are depicted by solid arrows, other are dashed.

Another problem which must be solved is dealing with English nodes that have an empty set of corresponding words, for example the English preposition *of*. In these cases, the algorithm goes directly to its children and counterparts of its children (the words *finanzpolitischer* and *Maßnahmen*) become children of the counterpart of its parent (the word *Koordination*). It means that the node *Maßnahmen* is a child of the node *Koordination*, even though there is one more node (*of*) between *policies* and *coordination*.

5 Data Filtering

When we have the dependency trees projected into the foreign language, we can simply train a parser on them and measure the parsing quality on some manually annotated treebank. The problem is that the quality of some trees on which we are training is very poor. This can be caused by various errors, mainly in preprocessing:

- *non-parallel sentences* – two parallel sentences have completely different meaning. This is caused by an error in sentence-level alignment. Sometimes it happens that only a part of a sentence is translated.
- *completely different structure* – the sentences have the same meaning but their syntactic structures are completely different.
- *wrong word alignment* – in case there are more words with very low frequency in the sentence.

We would like to filter out these bad sentences before training. For this purpose, we established two metrics of the sentence quality: *alignment sparseness* and *non-projectivity*.

5.1 Alignment Sparseness Limit

We define *alignment sparseness* as a relative number of words that have no counterpart in an *intersection* alignment. It is computed as a number of links divided by the average length of the pair of sentences. Its values are between 0 and 1. Value 0 means that the parallel sentences have the same length and there is a perfect 1-to-1 alignment mapping. Value 1 means that there are no intersection links at all.

$$S = 1 - \frac{\#links}{\frac{1}{2}(length(e) + length(f))}$$

All sentences that have higher *alignment sparseness* than a given threshold are filtered out. There is a trade-off between quality and quantity of the training data. Figure 8 shows the experiment in searching for optimal sparseness limit S_{max}. We can see that the higher the limit S_{max}, the higher the number of training sentences. If we train a parser on them and test it on a treebank, accuracy of such parser increases at first, but then it begins to decrease slightly, because the number of wrongly aligned sentences in the training data grows.

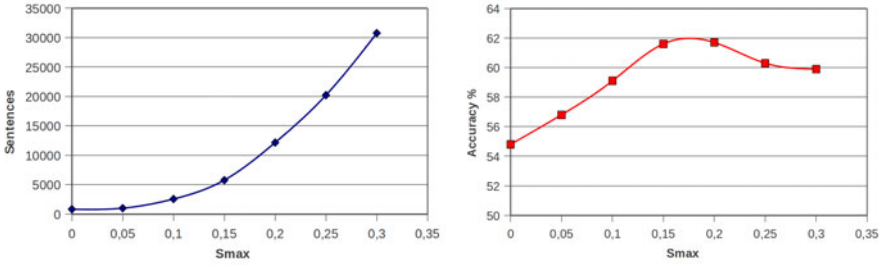

Fig. 8. Relations between alignment sparseness limit, number of sentences after filtering, and unlabeled parsing accuracy. This was measured on English-Czech News commentaries parallel corpus with approximately 100,000 sentence pairs. You can see that the optimal limit S_{max} here is 0.2. For this limit, the parser was trained only on 12,500 sentences, which means that we filter out more than 87% of sentences.

5.2 Non-projectivity Limit

The next criterion for recognition of trees that are not suitable for training the parser is the number of non-projective edges in them. An edge $[d, g]$ in a tree is non-projective, if the parent p of the governing node g lays between d and g. Of course, some non-projective edges can be correct, but after the review of the projected trees, we found out that a majority of non-projective edges are errors caused mainly by the wrong word alignment.

The experiment in which we filter out also such sentences where there were more non-projectivities than a given limit is shown in Figure 9. We can see that the best parsing accuracy was achieved by filtering out all sentences containing at least one non-projective edge (the limit $NP_{max} = 0$).

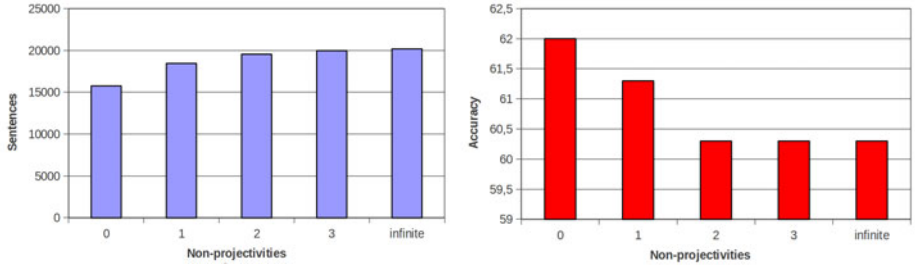

Fig. 9. Experiment with filtering out the sentences containing more non-projectivities than a given limit NP_{max}. It was done on Czech-English parallel corpus already filtered by alignment sparseness limit $S_{max} = 0.25$, which is more than 20,000 sentence pairs.

6 Experiments

We ran our experiments on four languages: Bulgarian, Czech, Dutch, and German. For Czech and German, we used the News commentaries parallel corpus as it was prepared for the WMT10 translation task.[1] For Bulgarian and Dutch, we used

[1] `http://www.statmt.org/wmt10/`

the Acquis Communautaire parallel corpus[2]. The English side of the corpus was tagged by Morce tagger [9] and parsed by McDonald's maximum spanning tree parser [10] which was trained on English CoNLL X[3] data. The foreign (target) sides were tagged by Tree-tagger [11] with appropriate models downloaded from TreeTagger websites.[4]

For filtering the projected trees, we tried several values for *alignment sparseness* limit and *non-projectivity* limit. Once we had the filtered trees, we trained the MST parser[5] on them. The parser was then tested on development-test data from the CoNLL X shared task [12]. The attachment accuracies are in Table 1.

Table 1. Unlabeled parsing accuracies for Bulgarian, Czech, Dutch, and German tested on CoNLL X testing data. S_{max} and NP_{max} are the thresholds used for filtering data before training. The "EN parser" column describes the post-processing steps used for English parsing.

Language	Parallel Corpus	EN parser	S_{max}	NP_{max}	Accuracy
Bulgarian	Acquis Communautaire	*CoNLL+CoordTr*	0.2	0	52.7 %
Czech	News commentaries	*CoNLL+AuxVTr*	0.15	0	62.0 %
Dutch	Acquis Communautaire	*CoNLL+CoordTr*	0.2	0	52.4 %
German	News commentaries	*CoNLL+CoordTr*	0.2	0	55.7 %

Ideally, the testing treebanks should follow the same annotation guidelines as the treebank on which we train English. However, that is not true in CoNLL X. The treebanks differ for example in capturing coordination structures or dealing with auxiliary verbs. For this reasons we implemented two post-processing steps; one for transforming coordination structures (*CoordTr*)[6] and the other for rehanging auxiliary verbs (*AuxVTr*).[7] The basic parsing (using the CoNLL data) is marked in Table 1 as *CoNLL*.

In order to compare our projection method with previous works, we ran the whole process on Bulgarian with exactly same setting as in [6]. We used the English-Bulgarian OpenSubtitles parallel corpus [13], the English side was parsed by McDonald's MST parser trained on sections 2-21 of the Penn treebank with dependencies extracted using the head rules of Yamada and Matsumoto [14]. The parser was tested on the Bultreebank corpus as it was released for CoNLL X [12]

[2] Only the one-to-one sentence pairs were extracted from the parallel corpus and due to the computability reasons, we used only the first 100,000 parallel sentences which length was higher than two and lower than 30 words.

[3] `http://nextens.uvt.nl/~conll/`

[4] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

[5] We used the McDonald's parser `mstparser-0.4.3b` with these settings: `order:1`, `iters:10`, `decode-type:proj`, `training-k:1`.

[6] In English CoNLL data, the head of coordination structure is the conjunction. However, in Bulgarian and German, the first coordinating member is the head and the other members and conjunctions become its children.

[7] Auxiliary verbs in Czech CoNLL data (*byl*, *bude*, *by*) depend on the main verb, while English auxiliary verbs (*do*, *will*, *be*, *have*) do not.

Table 2. Comparison of unlabeled parsing accuracies on Bulgarian CoNLL X training set. Our method is by 0.3% better than the method presented by Ganchev et al. in their work [6].

Method	Parser	Accuracy
Ganchev et al.	Discriminative model	66.9 %
Ganchev et al.	Generative Model	67.8 %
Our method	MST parser	68.1 %

shared task on the training sentences of up to 10 words. Punctuation was stripped at training time. The results compared in Table 2 show that our method outperforms the previous work [6] in unlabeled accuracy of the parser.

The projection algorithm written in Perl, example data, and the instructions how to run the whole process including syntactic analysis and alignment can be downloaded from `http://www.cicling.org/2011/software/49/`.

7 Conclusions and Future Plans

In this paper, we describe a novel method for projecting dependency trees across parallel texts, in which diverse word-alignment symmetrizations are used. Even though we do not combine the projected dependencies with automatically inferred dependencies, and we train MST parser directly on the projected trees without any modifications, we show that the parsing accuracies reached by our simple projection method are comparable to the previous more complex works. We have made the comparison on Bulgarian data and we outperform the previous state-of-the-art result by 0.3%.

We see two main advantages of the presented algorithm. First, it uses different types of alignment links and therefore some of them may be more preferred in the projection than others. Second, the *X-to-English* alignment ensures that all words on the target side are linked somewhere. This fact helps in attachment of function words which do not have their own counterpart in English.

The biggest problem of this task is differences in annotation guidelines of particular treebanks. This fact makes the evaluation problematic. We would like to solve it in the future by creating more rules to make the treebanks more similar or even create a small multilingual treebank with the same annotation rules for several languages, which would be very useful for evaluating such experiments.

We are also aware of great amounts of trees that are filtered out before training. In the future we plan to incorporate into training data all well-aligned subtrees, not only the whole sentences.

Acknowledgments

This work has been supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 247762 (FAUST, FP7-ICT-2009-4-247762), by the doctoral grant "Res Informatica" of the Grant Agency of the Czech Republic (GA201/09/H057), by grant GAUK 116310 of

the Grant Agency of Charles University, and by the grants MSM 0021620838 and SVV Projekt 261314/2010. We are grateful to the anonymous reviewers for further research suggestions.

References

1. Klein, D., Manning, C.D.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, ACL 2004. Association for Computational Linguistics, Morristown (2004)
2. Koo, T., Carreras, X., Collins, M.: Simple semi-supervised dependency parsing. In: Proceedings of ACL/HLT (2008)
3. Hwa, R., Resnik, P., Weinberg, A., Kolak, O.: Evaluating Translational Correspondence using Annotation Projection. In: Proceedings of the 40th Annual Meeting of the ACL, pp. 392–399 (2002)
4. Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., Kolak, O.: Bootstrapping Parsers via Syntactic Projection across Parallel Texts. Natural Language Engineering 11, 11–311 (2005)
5. Smith, D.A., Eisner, J.: Parser adaptation and projection with quasi-synchronous grammar features. In: EMNLP 2009: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 822–831. Association for Computational Linguistics, Morristown (2009)
6. Ganchev, K., Gillenwater, J., Taskar, B.: Dependency grammar induction via bitext projection constraints. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2009, pp. 369–377. Association for Computational Linguistics, Morristown (2009)
7. Jiang, W., Liu, Q.: Dependency parsing and projection based on word-pair classification. In: ACL 2010: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 12–20. Association for Computational Linguistics, Morristown (2010)
8. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1), 19–51 (2003)
9. Spoustová, D., Hajič, J., Votrubec, J., Krbec, P., Květoň, P.: The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In: ACL 2007: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing, pp. 67–74. Association for Computational Linguistics, Morristown (2007)
10. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-Projective Dependency Parsing using Spanning Tree Algorithms. In: Proceedings of Human Langauge Technology Conference and Conference on Empirical Methods in Natural Language Processing (HTL/EMNLP), Vancouver, BC, Canada, pp. 523–530 (2005)
11. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK, vol. 12, pp. 44–49 (1994)
12. Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of The Tenth Conference on Natural Language Learning (CoNLL-X), New York City, USA, pp. 149–164 (2006)
13. Tiedemann, J.: Building a Multilingual Parallel Subtitle Corpus. In: Proceedings of CLIN (2007)
14. Yamada, H., Matsumoto, Y.: Statistical Dependency Analysis with Support Vector Machines. In: Proceedings of IWPT, pp. 195–206 (2003)