Data-Driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data

Kathrin Spreyer and **Jonas Kuhn**
Department of Linguistics
University of Potsdam, Germany
{spreyer,kuhn}@ling.uni-potsdam.de

Abstract

We present a simple but very effective approach to identifying high-quality data in noisy data sets for structured problems like parsing, by greedily exploiting partial structures. We analyze our approach in an annotation projection framework for dependency trees, and show how dependency parsers from two different paradigms (graph-based and transition-based) can be trained on the resulting tree fragments. We train parsers for Dutch to evaluate our method and to investigate to which degree graph-based and transition-based parsers can benefit from incomplete training data. We find that partial correspondence projection gives rise to parsers that outperform parsers trained on aggressively filtered data sets, and achieve unlabeled attachment scores that are only 5% behind the average UAS for Dutch in the CoNLL-X Shared Task on supervised parsing (Buchholz and Marsi, 2006).

1 Introduction

Many weakly supervised approaches to NLP rely on heuristics or filtering techniques to deal with noise in unlabeled or automatically labeled training data, e.g., in the exploitation of parallel corpora for cross-lingual projection of morphological, syntactic or semantic information. While heuristic approaches can implement (linguistic) knowledge that helps to detect noisy data (e.g., Hwa et al. (2005)), they are typically task- and language-specific and thus introduce a component of indirect supervision. Non-heuristic filtering techniques, on the other hand, employ reliability measures (often unrelated to the task) to predict high-precision data points (e.g., Yarowsky et al. (2001)). In order to reach a sufficient level of precision, filtering typically has to be aggressive, especially for highly structured tasks like parsing. Such aggressive filtering techniques incur massive data loss and enforce trade-offs between the quality and the amount of usable data.

Ideally, a general filtering strategy for weakly supervised training of structured analysis tools should eliminate noisy subparts in the automatic annotation without discarding its high-precision aspects; thereby data loss would be kept to a minimum. In this paper, we propose an extremely simple approach to noise reduction which greedily exploits partial correspondences in a parallel corpus, i.e., correspondences potentially covering only substructures of translated sentences. We implemented this method in an annotation projection framework to create training data for two dependency parsers representing different parsing paradigms: The MST-Parser (McDonald et al., 2005) as an instance of *graph-based dependency parsing*, and the Malt-Parser (Nivre et al., 2006) to represent *transition-based dependency parsing*. In an empirical evaluation, we investigate how they react differently to incomplete and noisy training data.

Despite its simplicity, the partial correspondence approach proves very effective and leads to parsers that achieve unlabeled attachment scores that are only 5% behind the average UAS for Dutch in the CoNLL-X Shared Task (Buchholz and Marsi, 2006).

After a summary of related work in Sec. 2, we discuss dependency tree projection (Sec. 3) and partial correspondence (Sec. 4). In Sec. 5, we give an overview of graph- and transition-based dependency parsing and describe how each can be adapted for training on partial training data in Sec. 6. Experimental results are presented in Sec. 7, followed by an analysis in Sec. 8. Sec. 9 concludes.

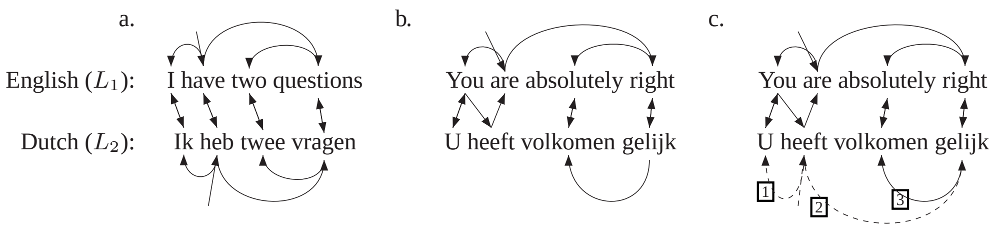


Figure 1: Dependency tree projection from English to Dutch. (a) Ideal scenario with bidirectional alignments. (b) Projection fails due to weak alignments. (c) Constrained fallback projection.

2 Related Work

Annotation projection has been applied to many different NLP tasks. On the word or phrase level, these include morphological analysis, part-of-speech tagging and NP-bracketing (Yarowsky et al., 2001), temporal analysis (Spreyer and Frank, 2008), or semantic role labeling (Padó and Lapata, 2006). In these tasks, word labels can technically be introduced in isolation, without reference to the rest of the annotation. This means that an aggressive filter can be used to discard unreliable data points (words in a sentence) without necessarily affecting high-precision data points in the same sentence. By using only the bidirectional word alignment links, one can implement a very robust such filter, as the bidirectional links are generally reliable, even though they have low recall for overall translational correspondences (Koehn et al., 2003). The bidirectional alignment filter is common practice (Padó and Lapata, 2006); a similar strategy is to discard entire sentences with low aggregated alignment scores (Yarowsky et al., 2001).

On the sentence level, Hwa et al. (2005) were the first to project dependency trees from English to Spanish and Chinese. They identify unreliable target parses (as a whole) on the basis of the number of unaligned or over-aligned words. In addition, they manipulate the trees to accommodate for non-isomorphic sentences. Systematic non-parallelisms between source and target language are then addressed by hand-crafted rules in a post-projection step. These rules account for an enormous increase in the unlabeled f-score of the direct projections, from 33.9 to 65.7 for Spanish and from 26.3 to 52.4 for Chinese. But they need to be designed anew for every target language, which is time-consuming and requires knowledge of that language.

Research in the field of unsupervised and weakly supervised parsing ranges from various forms of EM training (Pereira and Schabes, 1992; Klein and Manning, 2004; Smith and Eisner, 2004; Smith and Eisner, 2005) over bootstrapping approaches like self-training (McClosky et al., 2006) to feature-based enhancements of discriminative reranking models (Koo et al., 2008) and the application of semi-supervised SVMs (Wang et al., 2008). The partial correspondence method we present in this paper is compatible with such approaches and can be combined with other weakly supervised machine learning schemes. Our approach is similar to that of Clark and Curran (2006) who use partial training data (CCG lexical categories) for domain adaptation; however, they assume an existing CCG resource for the language in question to provide this data.

3 Projection of Dependency Trees

Most state-of-the-art parsers for natural languages are data-driven and depend on the availability of sufficient amounts of labeled training data. However, manual creation of treebanks is time-consuming and labour-intensive. One way to avoid the expensive annotation process is to automatically label the training data using *annotation projection* (Yarowsky et al., 2001): Given a suitable resource (such as a parser) in language L_1, and a word-aligned parallel corpus with languages L_1 and L_2, label the L_1-portion of the parallel text (with the parser) and copy the annotations to the corresponding (i.e., aligned) elements in language L_2. This is illustrated in Fig. 1a. The arrows between English and Dutch words indicate the word alignment. Assuming we have a parser to produce the dependency tree for the English sentence, we build the tree for the Dutch sentence by establishing arcs between words w_D (e.g., *Ik*) and h_D (*heb*) if there are aligned pairs (w_D, w_E)

	#sents w/ projected parse	avg. sent length	vocab (lemma)
unfiltered	(100,000)	24.92	19,066
bidirectional	2,112	6.39	1,905
fallback	6,426	9.72	4,801
bi+frags$_{\leq 3}$	7,208	9.44	4,631

Table 1: Data reduction effect of noise filters.

(*Ik* and *I*) and (h_D, h_E) (*heb* and *have*) such that h_E is the head of w_E in the English tree.

Annotation projection assumes *direct correspondence* (Hwa et al., 2005) between languages (or annotations), which—although it is valid in many cases—does not hold in general: non-parallelism between corresponding expressions in L_1 and L_2 causes errors in the target annotations. The word alignment constitutes a further source for errors if it is established automatically—which is typically the case in large parallel corpora.

We have implemented a language-independent framework for dependency projection and use the Europarl corpus (Koehn, 2005) as the parallel text. Europarl consists of the proceedings of the European Parliament, professionally translated in 11 languages (approx. 30mln words per language). The data was aligned on the word level with GIZA++ (Och and Ney, 2003).[1] In the experiments reported here, we use the language pair English-Dutch, with English as the source for projection (L_1) and Dutch as L_2. The English portion of the Europarl corpus was lemmatized and POS tagged with the TreeTagger (Schmid, 1994) and then parsed with MaltParser (which is described in Sec. 6), trained on a dependency-converted version of the WSJ part from the Penn Treebank (Marcus et al., 1994), but with the automatic POS tags. The Dutch sentences were only POS tagged (with TreeTagger).[2]

3.1 Data Loss Through Filtering

We quantitatively assess the impact of various filtering techniques on a random sample of 100,000 English-Dutch sentence pairs from Europarl (avg. 24.9 words/sentence). The English dependency trees are projected to their Dutch counterparts as explained above for Fig. 1a.

The first filter we examine is the one that considers exclusively bidirectional alignments. It admits dependency arcs to be projected only if the head h_E and the dependent w_E are each aligned *bidirectionally* with some word in the Dutch sentence. This is indicated in Fig. 1b, where the English verb *are* is aligned with the Dutch translation *heeft* only in one direction. This means that none of the dependencies involving *are* are projected, and the projected structure is not connected. We will discuss in subsequent sections how less restricted projection methods can still incorporate such data.

Table 1 shows the quantitative effect of the bidirectional filter in the row labeled 'bidirectional'. The proportion of usable sentences is reduced to 2.11%. Consequently, the vocabulary size diminishes by a factor of 10, and the average sentence length drops considerably from almost 25 to less than 7 words, suggesting that most non-trivial examples are lost.

3.2 Constrained Fallback Projection

As an instance of a more relaxed projection of complete structures, we also implemented a fallback to unidirectional links which projects further dependencies *after* a partial structure has been built based on the more reliable bidirectional links. That is, the dependencies established via unidirectional alignments are constrained by the existing subtrees, and are subject to the wellformedness conditions for dependency trees.[3] Fig. 1c shows how the fallback mechanism, initialized with the unconnected structure built with the bidirectional filter, recovers a parse tree for the weakly aligned sentence pair in Fig. 1b. Starting with the leftmost word in the Dutch sentence and its English translation (*U* and *You*), there is a unidirectional alignment for the head of *You*: *are* is aligned to *heeft*, so *U* is established as a dependent of *heeft* via fallback. Likewise, *heeft* can now be identified as the root node. Note that the (incorrect) alignment between *heeft* and *You* will not be pursued because it would lead to *heeft* being a dependent of itself and thus violating the wellformed-

[1]Following standard practice, we computed word alignments in both directions ($L_1 \rightarrow L_2$ and $L_2 \rightarrow L_1$); this gives rise to two unidirectional alignments. The *bidirectional alignment* is the intersection of the two unidirectional ones.

[2]The Dutch POS tags are used to train the monolingual parsers from the projected dependency trees (Sec. 7).

[3]I.e., single headedness and acyclicity; we do not require the trees to be projective, but instead train pseudo-projective models (Nivre and Nilsson, 2005) on the projected data (cf. fn. 5).

#frags	1	2	3	4–15	>15
#words					
<4	425	80	12	–	–
4–9	**1,331**	**1,375**	**1,567**	4,793	–
10–19	**339**	**859**	**1,503**	27,910	522
20–30	**17**	**45**	**143**	20,756	10,087
>30	**0**	**5**	**5**	4,813	23,362

Table 2: Fragmented parses projected with the alignment filter. The sentences included in the data set ‘bi+frags$_{\leq 3}$’ are in boldface.

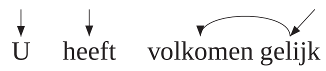
ness conditions. Finally, the subtree rooted in *gelijk* is incorporated as the second dependent of *heeft*.

As expected, the proportion of examples that pass this filter rises, to 6.42% (Table 1, ‘fallback’). However, we will see in Sec. 7 that parsers trained on this data do not improve over parsers trained on the bidirectionally aligned sentences alone. This is presumably due to the noise that inevitably enters the training data through fallback.

4 Partial Correspondence Projection

So far, we have only considered complete trees, i.e., projected structures with exactly one root node. This is a rather strict requirement, given that even state-of-the-art parsers sometimes fail to produce plausible complete analyses for long sentences, and that non-sentential phrases such as complex noun phrases still contain valuable, non-trivial information. We therefore propose *partial correspondence projection* which, in addition to the complete annotations produced by tree-oriented projection, yields partial structures: It admits fragmented analyses in case the tree-oriented projection cannot construct a complete tree. Of course, the nature of those fragments needs to be restricted so as to exclude data with no (interesting) dependencies. E.g., a sentence of five words with a parse consisting of five fragments provides virtually no information about dependency structure. Hence, we impose a limit (fixed at 3 after quick preliminary tests on automatically labeled development data) on the number of fragments that can make up an analysis. Alternatively, one could require a minimum fragment size.

As an example, consider again Fig. 1b. This example would be discarded in strict tree projection, but under partial correspondence it is included as a partial analysis consisting of three fragments:



Although the amount of information provided in this analysis is limited, the arc between *gelijk* and *volkomen*, which is strongly supported by the alignment, can be established without including potentially noisy data points that are only weakly aligned.

We use partial correspondence in combination with bidirectional projection.[4] As can be seen in Table 1 (‘bi+frags$_{\leq 3}$’), this combination boosts the amount of usable data to a range similar to that of the fallback technique for trees; but unlike the latter, partial correspondence continues to impose a high-precision filter (bidirectionality) while improving recall through relaxed structural requirements (partial correspondence). Table 2 shows how fragment size varies with sentence length.

5 Data-driven Dependency Parsing

Models for data-driven dependency parsing can be roughly divided into two paradigms: Graph-based and transition-based models (McDonald and Nivre, 2007).

5.1 Graph-based Models

In the graph-based approach, global optimization considers all possible arcs to find the tree $\hat{T}$ s.t.

$$\hat{T} = \arg\max_{T \in D} s(T) = \arg\max_{T \in D} \sum_{(i,j,l) \in A_T} s(i,j,l)$$

where D is the set of all well-formed dependency trees for the sentence, A_T is the set of arcs in T, and $s(i,j,l)$ is the score of an arc between words w_i and w_j with label l. The specific graph-based parser we use in this paper is the MSTParser of McDonald et al. (2005). The MSTParser learns the scoring function s using an online learning algorithm (Crammer and Singer, 2003) which maximizes the margin between $\hat{T}$ and $D \setminus \{\hat{T}\}$, based on a loss function that counts the number of words with incorrect parents relative to the correct tree.

5.2 Transition-based Models

In contrast to the global optimization employed in graph-based models, transition-based models construct a parse tree in a stepwise way: At each point,

[4]Fragments from fallback projection turned out not to be helpful as training data for dependency parsers.

the locally optimal parser action (*transition*) t^* is determined greedily on the basis of the current configuration c (previous actions plus local features):

$$t^* = \arg\max_{t \in T} \; s(c, t)$$

where T is the set of possible transitions. As a representative of the transition-based paradigm, we use the MaltParser (Nivre et al., 2006). It implements incremental, deterministic parsing algorithms and employs SVMs to learn the transition scores s.

6 Parsing with Fragmented Trees

To make effective use of the fragmented trees produced by partial correspondence projection, both parsing approaches need to be adapted for training on sentences with unconnected substructures. Here we briefly discuss how we represent these structures, and then describe how we modified the parsers.

We use the CoNLL-X data format for dependency trees (Buchholz and Marsi, 2006) to encode partial structures. Specifically, every fragment root specifies as its head an artificial root token w_0 (distinguished from a true root dependency by a special relation FRAG). Thus, sentences with a fragmented parse are still represented as a single sentence, including all words; the difference from a fully parsed sentence is that unconnected substructures are attached directly under w_0. For instance, the partial parse in Fig. 1b would be represented as follows (details omitted):

(1)
```
1 U        pron 0 FRAG
2 heeft    verb 0 ROOT
3 volkomen adj  4 mod
4 gelijk   noun 0 FRAG
```

6.1 Graph-based Model: fMST

In the training phase, the MSTParser tries to maximize the scoring margin between the correct parse and all other valid dependency trees for the sentence. However, in the case of fragmented trees, the training example is not strictly speaking correct, in the sense that it does not coincide with the desired parse tree. In fact, this desired tree is among the other possible trees that MST assumes to be incorrect, or at least suboptimal. In order to relax this assumption, we have to ensure that the loss of the desired tree is zero. While it is impossible to single out this one tree (since we do not know which one it is), we can steer the margin in the right direction with a loss function that assigns zero loss to all trees that are *consistent* with the training example, i.e., trees that differ from the training example at most on those words that are fragment roots (e.g., *gelijk* in Fig. 1). To reflect this notion of loss during optimization, we also adjust the definition of the score of a tree:

$$s(T) = \sum_{(i,j,l) \in A_T \colon l \neq \text{FRAG}} s(i, j, l)$$

We refer to this modified model as *f(iltering)MST*.

6.2 Transition-based Model: fMalt

In the transition-based paradigm, it is particularly important to preserve the original context (including unattached words) of a partial analysis, because the parser partly bases its decisions on neighboring words in the sentence.

Emphasis of the role of isolated FRAG dependents as context rather than proper nodes in the tree can be achieved, as with the MSTParser, by eliminating their effect on the margin learned by the SVMs. Since MaltParser scores local decisions, this simply amounts to suppressing the creation of SVM training instances for such nodes (*U* and *gelijk* in (1)). That is, where the feature model refers to context information, unattached words provide this information (e.g., the feature vector for *volkomen* in (1) contains the form and POS of *gelijk*), but there are no instances indicating how they should be attached themselves. This technique of excluding fragment roots during training will be referred to as *fMalt*.

7 Experiments

7.1 Setup

We train instances of the graph- and the transition-based parser on projected dependencies, and occasionally refer to these as “projected parsers”.[5]

All results were obtained on the held-out CoNLL-X test set of 386 sentences (avg. 12.9

[5]The MaltParsers use the projective Nivre arc-standard parsing algorithm. For SVM training, data are split on the coarse POS tag, with a threshold of 5,000 instances. MSTParser instances use the projective Eisner parsing algorithm, and first-order features. The input for both systems is projectivized using the head+path schema (Nivre and Nilsson, 2005).

	Malt	MST
Alpino	80.05	82.43
EP	75.33	73.09
Alpino + EP	77.47	81.63
baseline 1 (previous)	23.65	
baseline 2 (next)	27.63	

Table 3: Upper and lower bounds (UAS).

words/sentence) from the Alpino treebank (van der Beek et al., 2002). The Alpino treebank consists mostly of newspaper text, which means that we are evaluating the projected parsers, which are trained on Europarl, in an *out-of-domain* setting, in the absence of manually annotated Europarl test data.

Parsing performance is measured in terms of *unlabeled attachment score (UAS),* i.e., the proportion of tokens that are assigned the correct head, irrespective of the label.[6]

To establish upper and lower bounds for our task of weakly supervised dependency parsing, we proceed as follows. We train MaltParsers and MST-Parsers on (i) the CoNLL-X training portion of the Alpino treebank (195,000 words), (ii) 100,000 Europarl sentences parsed with the parser obtained from (i), and (iii) the concatenation of the data sets (i) and (ii). The first is a supervised upper bound (80.05/82.43% UAS)[7] trained on manually labeled in-domain data, while the second constitutes a weaker bound (75.33/73.09%) subject to the same out-of-domain evaluation as the projected parsers, and the third (77.47%) is a self-trained version of (i). We note in passing that the supervised model does not benefit from self-training. Two simple baselines provide approximations to a lower bound: Baseline 1 attaches every word to the preceding word, achieving 23.65%. Analogously, baseline 2 attaches every word to the following word (27.63%). These systems are summarized in Table 3.

[6]The labeled accuracy of our parsers lags behind the UAS, because the Dutch dependency relations in the projected annotations arise from a coarse heuristic mapping from the original English labels. We therefore report only UAS.

[7]The upper bound models are trained with the same parameter settings as the projected parsers (see fn. 5), which were adjusted for noisy training data. Thus improvements are likely with other settings: Nivre et al. (2006) report 81.35% for a Dutch MaltParser with optimized parameter settings. McDonald et al. (2006) report 83.57% with MST.

		words	Malt	MST
a.	trees (bidirectional)	13,500	65.94	67.76
	trees (fallback)	62,500	59.28	65.08
	bi+frags$_{\leq 3}$	68,000	55.09	57.14
	bi+frags$_{\leq 3}$ (fMalt/fMST)	68,000	**69.15**	**70.02**
b.	trees (bidirectional)	100,000	61.86	**69.91**
	trees (fallback)	100,000	60.05	64.84
	bi+frags$_{\leq 3}$	100,000	54.50	55.87
	bi+frags$_{\leq 3}$ (fMalt/fMST)	100,000	**68.65**	**69.86**
c.	trees (bidirectional)	102,300	63.32	69.85
	trees (fallback)	465,500	53.45	64.88
	bi+frags$_{\leq 3}$	523,000	51.48	57.20
	bi+frags$_{\leq 3}$ (fMalt/fMST)	523,000	**69.52**	**70.33**

Table 4: UAS of parsers trained on projected dependency structures for (a) a sample of 100,000 sentences, subject to filtering, (b) 10 random samples, each with 100,000 words after filtering (average scores given), and (c) the entire Europarl corpus, subject to filtering.

7.2 Results

Table 4a summarizes the results of training parsers on the 100,000-sentence sample analyzed above. Both the graph-based (MST) and the transition-based (Malt) parsers react similarly to the more or less aggressive filtering methods, but to different degrees. The first two rows of the table show the parsers trained on complete trees ('trees (bidirectional)' and 'trees (fallback)'). In spite of the additional training data gained by the fallback method, the resulting parsers do not achieve higher accuracy; on the contrary, there is a drop in UAS, especially in the transition-based model (-6.66%). The increased level of noise in the fallback data has less (but significant)[8] impact on the graph-based counterpart (-2.68%).

Turning to the parsers trained on partial correspondence data ('bi+frags$_{\leq 3}$'), we observe even greater deterioration in both parsing paradigms if the data is used as is. However, in combination with the fMalt/fMST systems ('bi+frags$_{\leq 3}$ (fMalt/fMST)'), both parsers significantly outperform the tree-

[8]Significance testing ($p<.01$) was performed by means of the t-test on the results of 10 training cycles (Table 4c 'trees (fb.)' only 2 cycles due to time constraints). For the experiments in Table 4a and 4c, the cycles differed in terms of the order in which sentences where passed to the parser. In Table 4b we base significance on 10 true random samples for training.

		Recall					Precision				
	dep. length	1	2	3–6	≥ 7	root	1	2	3–6	≥ 7	root
a.	trees (bi.)	83.41	66.44	52.94	40.64	52.45	82.46	66.06	61.38	34.95	50.97
	trees (fb.)	82.20	64.21	54.59	37.95	55.72	82.64	61.41	54.39	31.96	68.55
	bi+frags$_{\leq 3}$	70.18	59.50	46.61	32.14	**61.87**	**83.75**	67.22	58.25	32.81	27.01
	bi+frags$_{\leq 3}$ (fMalt)	**89.23**	**75.34**	**59.18**	41.65	59.06	**83.46**	69.05	**65.85**	**48.21**	**75.79**
	Alpino-Malt	92.81	84.94	75.11	65.44	66.15	89.71	81.08	77.56	62.57	84.58
b.	trees (bi.)	**87.53**	**73.79**	59.57	46.79	71.01	86.43	74.08	64.78	45.17	**66.79**
	trees (fb.)	82.53	69.37	55.77	37.46	70.24	85.31	69.29	59.85	40.14	53.99
	bi+frags$_{\leq 3}$	68.11	57.48	34.30	13.00	**90.68**	**90.28**	**78.54**	66.36	43.70	23.41
	bi+frags$_{\leq 3}$ (fMST)	**87.73**	72.84	**62.55**	**50.15**	67.78	86.94	71.60	66.05	**48.48**	**68.20**
	Alpino-MST	94.13	86.60	76.91	65.14	71.60	91.76	82.49	76.23	71.96	85.38

Table 5: Performance relative to dependency length. (a) Projected MaltParsers and (b) projected MSTParsers.

oriented models ('trees (bidirectional)') by 3.21% (Malt) and 2.26% (MST).

It would be natural to presume that the superiority of the partial correspondence filter is merely due to the amount of training data, which is larger by a factor of 5.04. We address this issue by isolating the effect on the quality of the data, and hence the success at noise reduction: In Table 4b, we control for the amount of data that is effectively used in training, so that each filtered training set consists of 100,000 words. Considering the Malt models, we find that the trends suggested in Table 4a are confirmed: The pattern of relative performance emerges even though any quantitative (dis-)advantages have been eliminated.[9] [10] Interestingly, the MSTParser does not appear to gain from the increased variety (cf. Table 1) in the partial data: it does not differ significantly from the 'trees (bi.)' model.

Finally, Table 4c provides the results of training on the entire Europarl, or what remains of the corpus after the respective filters have applied. The results corroborate those obtained for the smaller samples.

In summary, the results support our initial hypothesis that partial correspondence for sentences containing a highly reliable part is preferable to relaxing the reliability citerion, and—in the case of the transition-based MaltParser—also to aggressively filtering out all but the reliable complete trees. With UASs around 70%, both systems are only 5% behind the average 75.07% UAS achieved for Dutch in the CoNLL-X Shared Task.

8 Analysis

We have seen that the graph- and the transition-based parser react similarly to the various filtering methods. However, there are interesting differences in the magnitude of the performance changes. If we compare the two tree-oriented filters 'trees (bi.)' and 'trees (fb.)', we observe that, although both Malt and MST suffer from the additional noise that is introduced via the unidirectional alignments, the drop in accuracy is much less pronounced in the latter, graph-based model. Recall that in this paradigm, optimization is performed over the entire tree by scoring edges independenly; this might explain why noisy arcs in the training data have only a negligible impact. Conversely, the transition-based MaltParser, which constructs parse trees in steps of locally optimal decisions, has an advantage when confronted with partial structures: The individual fragments provide exactly the local context, plus lexical information about the (unconnected) wider context.

To give a more detailed picture of the differences between predicted and actual annotations, we show the performance (of the parsers from Table 4b) separately for binned arc length (Table 5) and sentence length (Table 6). As expected, the performance of both the supervised upper bounds (Alpino-

[9]The degree of skewedness in the filtered data is not controlled, as it is an important characteristic of the filters.

[10]Some of the parsers trained on the larger data sets (Table 4b+c) achieve worse results than their smaller counterparts in Table 4a. We conjecture that it is due to the thresholded POS-based data split, performed prior to SVM training: Larger training sets induce decision models with more specialized SVMs, which are more susceptible to tagging errors. This could be avoided by increasing the threshold for splitting.

	sent. length	<4	4–9	10–19	20–30	> 30
a.	trees (bi.)	**73.87**	62.13	65.67	60.81	55.18
	trees (fb.)	69.91	57.84	62.29	60.04	55.47
	bi+frags$_{\leq 3}$	74.14	54.40	56.62	54.07	48.95
	bi+fr$_{\leq 3}$ (fMalt)	**73.51**	**65.69**	**71.70**	**68.49**	**63.71**
	Alpino-Malt	81.98	69.81	81.11	82.82	76.02
b.	trees (bi.)	**76.67**	**70.16**	**73.09**	**69.56**	**63.57**
	trees (fb.)	73.24	64.93	67.79	64.98	57.70
	bi+frags$_{\leq 3}$	77.48	59.65	55.96	55.27	52.74
	bi+fr$_{\leq 3}$ (fMST)	73.24	**67.84**	**73.46**	**70.04**	**62.92**
	Alpino-MST	81.98	72.24	85.10	83.86	78.51

Table 6: UAS relative to sentence length. (a) Projected MaltParsers and (b) projected MSTParsers.

Malt/MST) and the projected parsers degrades as dependencies get longer, and the difference between the two grows. Performance across sentence length remains relatively stable. But note that both tables again reflect the pattern we saw in Table 4. Importantly, the relative ranking (in terms of f-score, not shown, resp. UAS) is still in place even in long distance dependencies and long sentences. This indicates that the effects we have described are not artifacts of a bias towards short dependencies.

In addition, Table 5 sheds some light on the impact of fMalt/fMST in terms of the trade-off between precision and recall. Without the specific adjustments to handle fragments, partial structures in the training data lead to an immense drop in recall. By contrast, when the adapted parsers fMalt/fMST are applied, they boosts recall back to a level comparable to or even above that of the tree-oriented projection parsers, while maintaining precision. Again, this effect can be observed across all arc lengths, except arcs to root, which naturally the 'bi+frags' models are overly eager to predict.

Finally, the learning curves in Fig. 2 illustrate how much labeled data would be required to achieve comparable performance in a supervised setting. The graph-based upper bound (Alpino-MST) reaches the performance of fMST (trained on the entire Europarl) with approx. 25,000 words of manually labeled treebank data; Alpino-Malt achieves the performance of fMalt with approx. 35,000 words. The manual annotation of even these moderate amounts of data involves considerable efforts, including the creation of annotation guidelines and tools, the training of annotators etc.

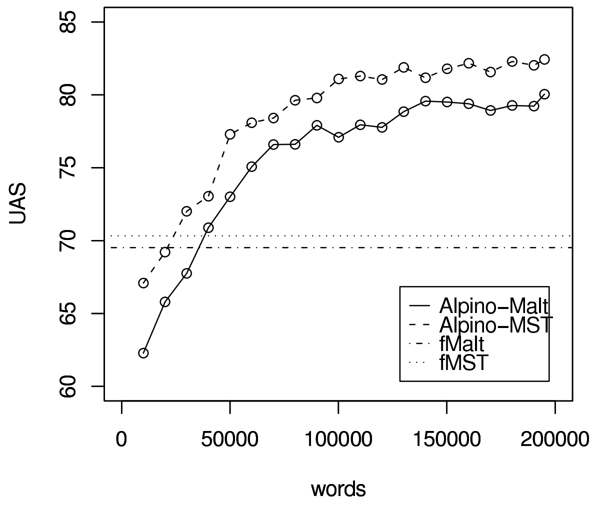


Figure 2: Learning curves for the supervised upper bounds. They reach the performance of the projected parsers with ∼25,000 (MST) resp. 35,000 (Malt) words.

9 Conclusion

In the context of dependency parsing, we have proposed partial correspondence projection as a greedy method for noise reduction, and illustrated how it can be integrated with data-driven parsing. Our experimental results show that partial tree structures are well suited to train transition-based dependency parsers. Graph-based models do not benefit as much from additional partial structures, but instead are more robust to noisy training data, even when the training set is very small.

In future work, we will explore how well the techniques presented here for English and Dutch work for languages that are typologically further apart, e.g., English-Greek or English-Finnish. Moreover, we are going to investigate how our approach, which essentially ignores unknown parts of the annotation, compares to approaches that marginalize over hidden variables. We will also explore ways of combining graph-based and transition-based parsers along the lines of Nivre and McDonald (2008).

Acknowledgments

The research reported in this paper has been supported by the German Research Foundation DFG as part of SFB 632 "Information structure" (project D4; PI: Kuhn).

References

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164, New York City, June.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of HLT-NAACL 2006*, pages 144–151, New York, June.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Reseach*, 3:951–991, January.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 2004*, pages 478–485, Barcelona, Spain.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the MT Summit 2005*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-HLT 2008)*, pages 595–603, Columbus, Ohio, June.

Mitchell Marcus, Grace Kim, Mary Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL 2006*, pages 152–159, New York, June.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL 2007*, pages 122–131.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP 2005)*.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL-X*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-HLT 2008*, pages 950–958, Columbus, Ohio, June.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*, pages 99–106.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL-X*, pages 221–225.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of COLING/ACL 2006*, Sydney, Australia.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL 1992*, pages 128–135.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, England.

Noah A. Smith and Jason Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proceedings of ACL 2004*, pages 487–494, Barcelona, July.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL 2005*, pages 354–362, Ann Arbor, MI, June.

Kathrin Spreyer and Anette Frank. 2008. Projection-based acquisition of a temporal labeller. In *Proceedings of IJCNLP 2008*, Hyderabad, India, January.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-HLT 2008*, pages 532–540, Columbus, Ohio, June.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001*.