Parsing Natural Language Sentences by Semi-supervised Methods

Rudolf Rosa
Institute of Formal and Applied Linguistics
Charles University in Prague, Faculty of Mathematics and Physics
Malostranské náměstí 25, Prague, Czech Republic
rosa@ufal.mff.cuni.cz

Abstract

We present our work on semi-supervised parsing of natural language sentences, focusing on multi-source crosslingual transfer of delexicalized dependency parsers. We first evaluate the influence of treebank annotation styles on parsing performance, focusing on adposition attachment style. Then, we present KL_{cpos^3}, an empirical language similarity measure, designed and tuned for source parser weighting in multi-source delexicalized parser transfer. And finally, we introduce a novel resource combination method, based on interpolation of trained parser models.

1 Introduction

The problem of supervised dependency parsing of natural language sentences has been intensively studied for the past decade, especially since the invention of the graph-based MSTParser by McDonald et al. (2005a), and the transition-based Malt parser by Nivre et al. (2006). The success of these parsing algorithms, together with several CoNLL shared tasks focused on dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007; Surdeanu et al., 2008; Hajič et al., 2009), even lead to a general transition from constituency parsing to dependency parsing throughout the NLP community. The current state-of-the-art dependency parsers, such as the Mate parser of Bohnet and Nivre (2012), often achieve around 90% UAS (Unlabelled Attachment Score) for many languages.

The supervised parsing approaches require labelled training data, i.e., manually created dependency treebanks. While these are available for dozens of languages (see Section 2), only around 1% of the world's languages are covered by treebanks. Moreover, treebank annotation is costly, and it is not expected that most of the remaining languages will be processed any time soon, or ever. To make matters worse, the existing treebanks necessarily capture texts from limited domains and limited time periods only, and do not serve us well when we need to parse texts from a different domain, as shown e.g. by Gildea (2001). This naturally motivates research of semi-supervised or unsupervised parsing methods.

In our work, we focus on semi-supervised approaches to the multilinguality issue, investigating the possibilities of using the knowledge contained in treebanks for one or more source languages (*src*) to analyze sentences of a different target language (*tgt*).[1] Specifically, we perform transfer of delexicalized dependency parsers – see Section 3, in which we review the existing approaches.

As noted in Section 2, a plethora of treebank annotation styles exist, and it is not entirely clear how the annotation style relates to parser performance. It is well-known that some annotation styles are more easily learned by dependency parsers than other, but the research in this area is rather rudimentary even for the monolingual supervised setting, and practically non-existent in other areas, including cross-lingual parser transfer. As a prominent example of an annotation difference known to be important for parser accuracy, but also strongly influencing cross-lingual annotation coherence (with opposing effects), in Section 4, we thoroughly study the appropriateness of two adposition annotation styles, Prague and Stanford, for delexicalized parser transfer.

Linguistic intuition tells us that for cross-lingual parser transfer, using a *src* treebank of a language

[1] While our motivation is the analysis of languages without treebanks, we only evaluate our methods on languages for which treebanks are available, and simulate the under-resourced setting by not using the *tgt* treebanks for training. This is a natural consequence of the fact that without a test treebank, intrinsic evaluation is impossible, and we are not aware of any reliable scenario for extrinsic parser evaluation.

very close to the *tgt* language should bring the best results. However, as shown by McDonald et al. (2011), not only is the similarity of languages only a weakly established concept, but the empirical results are often rather counter-intuitive – for example, to parse Swedish, the best treebank to use turned out to be a Portuguese one, performing better than treebanks for Germanic languages (their dataset included, among other, German, Dutch, Danish, and English). Therefore, in Section 5, we introduce a new empirical language similarity measure, designed and tuned specifically for the delexicalized parser transfer approaches, and evaluate its performance in several settings.

Furthermore, in Section 6, we introduce our own novel method of multisource delexicalized parser transfer, based on interpolation of trained parser models. We evaluate the method both in an unweighted as well as a weighted setting, and compare it to the standard resource combination methods.

Finally, in Section 7, we present the intention to enrich our approach by semi-supervised lexicalization in future, which other authors have already shown to have a great potential of improving the cross-lingual parser transfer performance.

2 Data

One of the positive side-effects of the CoNLL shared tasks was the assembly of dependency treebanks for many languages, usually simply referred to as the CoNLL treebanks, as well as the definition of a file format for representing parsed sentences – the CoNLL format. The datasets were used for evaluation in the shared tasks, but have also become the de-facto standard for evaluation of later dependency parsers, ensuring strong comparability of the reported results.

However, the CoNLL treebanks generally use different annotation styles on both morphological and syntactic level. For example, all treebanks define a POS (part of speech) tag for each word (or token, more precisely), but the set of POS tags used by each treebank is different, not only in the level of detail, but also in the actual tags used to carry the same information – a noun can be tagged as n, N, NN, No, S, IZE, etc. On the syntactic level, not only the sets of labels are different, but even the unlabelled dependency structures differ, as they correspond to different linguistic theories; probably the highest variance can be found in the annotation of coordination structures, as studied by Popel et al. (2013). While some of the differences may be motivated by inherent properties of the respective languages, they very often correspond merely to more-or-less arbitrary design decisions of technical rather than linguistic nature, taken during the creation of the treebanks. Importantly, such differences constitute unnecessary obstacles in most multilingual experiments. For cross-lingual parser transfer, these are absolutely crucial, leading to low performance of some methods and inapplicability of other.

The issues with cross-lingually incoherent annotation first led Zeman (2008) to the development of the Interset, a method of capturing values of most morphological features and for conversions between various tagsets. Later, the HamleDT collection of dependency treebanks was created by Zeman et al. (2012), consisting of treebanks harmonized not only on the morphological level (via Interset), but also on the syntactic level, loosely following the annotation style of the Prague Dependency Treebank of Böhmová et al. (2003).

In parallel, Petrov et al. (2012) defined the Universal POS tagset (UPOS) as a counter-weight to Interset, as it only captures 12 most important values of coarse-grained POS tags, ignoring all other morphological annotation. It was later used for annotation of the (eventually) 11 treebanks of the Google Universal Dependency Treebank collection of McDonald et al. (2013). For syntactic annotation, the authors defined their own version of the Stanford Dependencies (De Marneffe and Manning, 2008), modified to better suit the multilingual setting, as the original annotation style was implicitly designed for English. In turn, de Marneffe et al. (2014) reacted by introducing the Universal Stanford Dependencies as the "official" multilingual version of Stanford Dependencies. This annotation style was immediately adopted by Rosa et al. (2014), who modified it slightly and used it to "stanfordize" the HamleDT collection by implementing a language-neutral conversion pipeline and applying it to the harmonized treebanks in HamleDT 2.0.

Recently, all of the harmonization forces have joined together into the Universal Dependencies project of Nivre et al. (2015),[2] both defining an annotation style based mainly on UPOS, Interset

[2] `http://universaldependencies.github.io/docs/`

		Size (kTokens)	
	Language	Train	Test
ar	Arabic	250	28
bg	Bulgarian	191	6
bn	Bengali	7	1
ca	Catalan	391	54
cs	Czech	1,331	174
da	Danish	95	6
de	German	649	33
el	Greek	66	5
en	English	447	6
es	Spanish	428	51
et	Estonian	9	1
eu	Basque	138	15
fa	Persian	183	7
fi	Finnish	54	6
grc	Ancient Greek	304	6
hi	Hindi	269	27
hu	Hungarian	132	8
it	Italian	72	6
ja	Japanese	152	6
la	Latin	49	5
nl	Dutch	196	6
pt	Portuguese	207	6
ro	Romanian	34	3
ru	Russian	495	4
sk	Slovak	816	86
sl	Slovenian	29	7
sv	Swedish	192	6
ta	Tamil	8	2
te	Telugu	6	1
tr	Turkish	66	5

Table 1: List of HamleDT 2.0 treebanks.

and Universal Stanford Dependencies, as well as producing a set of 10 treebanks annotated in this way in the 1.0 version. More treebanks should be available soon, as the 1.1 version is due on 15th May 2015, and there is a firm plan on continuing to release more treebanks and to update the annotation style as appropriate in future. Thus, Universal Dependencies have the ambition of eventually becoming the ultimate annotation style and dataset for dependency parsing.

In our work, we carry out all experiments using HamleDT 2.0, as it is currently still the largest and most diverse harmonized treebank collection, consisting of 30 treebanks – see Table 1. Specifically, we use the stanfordized version of the collection for most experiments,[3] and the gold-standard UPOS tags[4] in all experiments. We always use the training sections of the treebanks to train the parsers or to estimate language similarities, and the test section to evaluate the methods.

We used 12 of the treebanks as a development set for hyperparameter tuning where appropriate to avoid overfitting to the dataset. The development set consisted of treebanks for Arabic, Bulgarian, Catalan, Greek, Spanish, Estonian, Persian, Finnish, Hindi, Hungarian, Italian, and Japanese;[5] the remaining 18 treebanks constitute the test set. For experiments where hyperparameter tuning on the development set was employed, we report the results of our methods separately for the test set and for the development set as *tgt* treebanks. However, for each *tgt* treebank, be it a test treebank or a development treebank, all remaining 29 *src* treebanks are always used for training in the evaluation of the methods.

Interestingly, the results of our methods results usually turned out to be generally similar or better on the test set than on the development set, suggesting that no overfitting happened. Therefore, we usually discuss both the results on the test set as well as on the development set when evaluating our experiments.

3 Delexicalized Parser Transfer

In the task of delexicalized dependency parser transfer, or delex transfer for short, we train a parser on a treebank for a resource-rich *src* language, using non-lexical features, most notably POS tags, but not using word forms or lemmas. Then, we apply that parser to POS-tagged sentences of a *tgt* language, to obtain a dependency parse tree. Delexicalized transfer yields worse results than a fully supervised lexicalized parser, trained on a treebank for the target language. However, for languages with no treebanks available, it may be useful to obtain at least a lower-quality parse tree for tasks such as information retrieval..

The idea of delexicalized transfer was conceived by Zeman and Resnik (2008), who trained a delexicalized parser on a Danish treebank and

[3] We chose the Stanford style conversion, instead of the HamleDT-native Prague style version, because Stanford Dependencies were developed with the objective of cross-lingual consistency of dependency structures. Thus, we expected them to perform better than other formalisms in cross-lingual experiments. Later evaluation of that decision, presented (non-chronologically) in Section 4, showed this assumption to be incorrect, but we have not redone all our experiments yet with respect to that finding.

[4] More precisely, the tags had been automatically converted from original gold-standard tags into UPOS tagset with Interset by the authors of HamleDT.

[5] To tune our methods to perform well in many different situations, we chose the development set to contain both smaller and larger treebanks, a pair of very close languages (ca, es), a very solitary language (ja), multiple members of several language families (Uralic, Romance), and both primarily left-branching (bg, el) and right-branching (ar, ja) languages.

evaluated it on a Swedish one. They note that while the lexicon of two languages will most probably differ significantly even if they are very close, they may share many morphological and syntactic properties. As a prerequisite to applying the method, they map the treebank POS tagsets to a common set, an approach later becoming known as conversion to Interset (Zeman, 2008). They also normalize the annotation styles of the treebanks to make them more similar, performing rule-based transformations – a method that has developed significantly since then and became known as treebank harmonization (Zeman et al., 2012). We largely build upon all of these approaches in our work.

Usually, multiple *src* treebanks are available, and it is non-trivial to select the best one for a given *tgt* language. Therefore, information from some or all *src* treebanks is usually combined together. The standard ways are to train a parser on the concatenation of all *src* treebanks (Section 3.2), or to train a separate parser on each *src* treebank and to combine the parse trees produced by the parsers using a maximum spanning tree algorithm (Section 3.3). The tree combination method typically performs better; it can also be easily extended by weighting the *src* parser predictions by similarity of the *src* language to the *tgt* language, which can further improve its results.

3.1 MSTperl parser

Throughout this work, we use the MSTperl parser of Rosa (2014), an implementation of the unlabelled single-best MSTParser of McDonald et al. (2005b), with first-order features and non-projective parsing. We train the parser using 3 iterations of MIRA (Crammer and Singer, 2003).

The MSTParser model uses a set of binary features F that are assigned weights w_f by training on a treebank. When parsing a sentence, the parser constructs a complete weighted directed graph over the tokens of the input sentence, and assigns each edge e a score s_e which is the sum of weights of features that are active for that edge:

$$s_e = \sum_{\forall f \in F} f(e) \cdot w_f \,. \tag{1}$$

The sentence parse tree is the maximum spanning tree (MST) over that graph, found using the algorithm of Chu and Liu (1965) and Edmonds (1967).

Our lexicalized feature set is based on (McDonald et al., 2005a), and consists of various conjunctions of the following features:

POS tags We use the coarse 12-value UPOS of Petrov et al. (2012).[6] For an edge, we use information about the POS tag of the head, dependent, their neighbours, and all of the nodes between them.

Token distance We use signed distance of head and dependent $(order_{head} - order_{dependent})$, bucketed into the following buckets:
$+1$; $+2$; $+3$; $+4$; $\geq +5$; $\geq +11$;
-1; -2; -3; -4; ≤ -5; ≤ -11.

Lexical features We use the word form and the morphological lemma of the head and the dependent.

The delexicalized feature set is based on the lexicalized one, but without the lexical features.

The usage of only this parser in all experiments somewhat limits the extent of our findings. Therefore, we intend to employ other parsers in future, e.g. the Malt parser of Nivre et al. (2006). For most of our approaches, this will be straightforward, but for the parser model interpolation approach (Section 6), it may be rather intriguing.

3.2 Treebank concatenation

McDonald et al. (2011) applied delexicalized transfer in a setting with multiple *src* treebanks available, finding that a treebank for a language that is typologically close to the *tgt* language is typically a good choice for the source treebank, but noting that the problem of selecting the best source treebank is non-trivial; we will further refer to the best source treebank as the *oracle treebank*, since it can hardly be identified without having a *tgt* language treebank for evaluation. As a workaround, the authors proposed a simple resource combination method – treebank concatenation – which consists of the following steps:

1. Concatenate all *src* treebanks.
2. Train a delex parser on the resulting treebank.
3. Apply the parser to the *tgt* text.

Applying this method led to better results than the average over individual single-source parsers, but worse than using only the oracle *src* parser. In our work, we take the treebank concatenation method as a baseline.

[6] These 12 values are: NOUN, VERB, ., ADJ, ADP, PRON, CONJ, ADV, PRT, NUM, DET, X.

3.3 Parse tree combination

An alternative resource combination approach is the parse tree combination, used by Sagae and Lavie (2006) for monolingual parser combination. In this approach, several independent parsers are applied to the same input sentence, and the parse trees they produce are combined into one resulting tree. The combination is performed using the idea of McDonald et al. (2005a), who formulated the problem of finding a parse tree as a problem of finding the maximum spanning tree of a weighted directed graph of potential parse tree edges. In the tree combination method, the weight of each edge is defined as the number of parsers which include that edge in their output (it can thus also be regarded as a parser voting approach). To find the MST, we use the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which was used by McDonald et al. (2005b) in the non-projective MSTParser. Other MST algorithms could be used, such as the Eisner algorithm (Eisner, 1996), which is, unlike Chu-Liu-Edmonds, constrained to producing only projective parse trees, and was used by McDonald et al. (2005a) in the projective MSTParser.

The tree combination method can be easily ported from a monolingual to a multilingual setting, where the individual parsers are trained over different languages. We take the tree combination method for our base approach to multi-source transfer, as it yields better results on average than the treebank concatenation method – probably because in treebank concatenation, larger treebanks have more influence on the result, which may not be well substantiated.

A nice feature of the tree combination approach is the straightforward possibility of assigning *weights* to the individual parsers, as done by Surdeanu and Manning (2010) in a monolingual setting. They let each parser contribute with a weight based on its performance (attachment score), thus giving a more powerful vote to parsers that seem to be better on average. While this is sensible in a monolingual setting, in multi-source delexicalized transfer we are more interested in the *language similarity* of the source and target language, as we would like to give more power to parsers trained on closer languages (see Section 5).

The parse tree combination method proceeds in the following way:

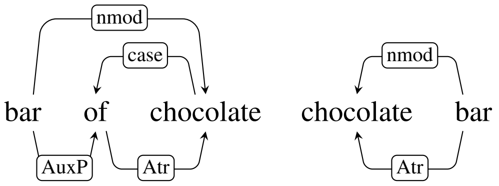

Figure 1: Stanford style (above) and Prague style (below) analysis of the phrases “bar of chocolate” and “chocolate bar”. Note that in Stanford style, these phrases have a more similar structure, both featuring an *nmod* edge from “bar” to “chocolate”. This shows the principle of constructions with a similar meaning also having a similar dependency structure.

1. Train a delex parser on each src treebank.
2. Apply each of the parsers to the tgt sentence, obtaining a set of parse trees.
3. Construct a weighted directed graph over tgt sentence tokens, with each edge assigned a score equal to the number of parse trees that contain this edge (i.e., each parse tree contributes by 0 or 1 to the edge score). In the weighted variant, the contribution of each src parse tree is multiplied by a weight $w(tgt, src)$, based on language similarity of tgt and src.
4. Find the maximum spanning tree over the graph with the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).

4 Treebank Annotation Style for Parsing

One of the prominent features of the newest versions of Stanford style dependencies is their approach to function words. The general rule is that all function words, such as adpositions or conjunctions, are attached as leaf nodes. This is a result of a lexicalist view of syntax, which favours direct dependency relations between lexical nodes directly, not mediated by function words. This also makes dependency structures more similar cross-lingually, as it is very common that the same function is expressed by an adposition in one language, but by other means, such as morphology or word order, in another language – or even within the same language, as shown in Figure 1.

The Prague style dependencies, on the other hand, are based upon a functionalist approach of Sgall (1967), and annotate adpositions as heads of adpositional groups. The lexical nodes are only directly connected in tectogrammatical (deepsyntax) dependency trees, where function words are removed and their functions are captured via node attributes. It is worth noting that in general, there is little difference between representing information by means of node attributes or leaf nodes; thus, Stanford trees and Prague tectogrammatical trees are actually very similar in structure. However, Prague tectogrammatical trees are rarely directly used in parsing – they are typically obtained either by manual annotation, or by automatic conversion from surface-syntax (analytical) trees.

While Stanford style trees may be more useful for further processing in NLP applications, it has been argued that Prague style trees are easier to obtain by using statistical parsers, as, among other differences, adpositions provide important cues to the parser for adpositional group attachment. This information becomes harder to access when the adpositions are annotated as leafs. The issue of dependency representation learnability has been studied by several authors, generally reaching similar conclusions (Schwartz et al., 2012; Søgaard, 2013; Ivanova et al., 2013). The approach suggested by de Marneffe et al. (2014) is to use a different annotation style for parsing, with Prague style adposition annotation, among other, and to convert the dependency trees to full Stanford style only after parsing.

Still, while this seems to be sufficiently proven in the general case, in multi-lingual parsing scenarios, the higher cross-lingual similarity of Stanford style dependency trees may be of benefit. From all of the differences between Prague and Stanford, the adposition attachment seems to be the most important, as adpositions are usually very frequent and diverse in languages, as well as very important in parsing. Therefore, in this section, we evaluate the influence of adposition annotation style in cross-lingual multi-source delexicalized parser transfer.

In Section 4.1, we show that the stanfordized version of HamleDT performs much worse for parsing than the Prague version. Consequently, in the following subsections, we use only the Prague version as the basis for our experiments, only employing on one of the prominent features of Stanford Dependencies – the adposition attachment. The other annotation differences are currently of less interest for us, as they concern less frequent phenomena and/or do not seem so promising for cross-lingual experiments. Thus, we alternate between Prague adposition attachment as head (denoted “P”), and Stanford adposition attachment as leaf node (denoted “S”), and thoroughly evaluate the effect of these annotation styles, with a focus on multi-source delexicalized parser transfer by parse tree combination.

Setup	Lex	Delex	Transfer
Prague	80.54	74.12	56.68
Stanford full	76.47	69.53	48.91
Prague non-punct	80.23	74.00	56.08
Stanford full non-punct	76.84	70.66	50.15

Table 2: Prague versus full Stanford annotation style, UAS averaged over 30 target languages. The *Lex*icalized and *Delex*icalized parser is monolingual; the *Transfer* parser is a combination of 29 parsers trained on source treebanks, and then applied to the remaining target language. Also lists UAS measured only on non-punctuation nodes.

4.1 Full Universal Stanford Dependencies

As a preliminary experiment, we compared the Prague version with its fully stanfordized version. The results are shown in Table 2. It can be seen that the Stanford version performs much worse than the Prague one – its results are lower by around 5% UAS absolute.

Closer inspection showed that many of the errors are actually due to sentence-final punctuation attachment. In Stanford style, sentence-final punctuation is to be attached as a dependent node of the root node of the sentence (typically the main predicate). However, this is difficult for the first-order parser, as it has no knowledge of the root node when scoring the potential edges, and thus the punctuation gets often attached to a different verb. In Prague style, the sentence-final punctuation is attached to the technical root node, which is marked by special values of the node features, and thus the assignment is very easy to make. While this is an important point to keep in mind when parsing into full Stanford style, it is of little relevance to the goal of this paper – punctuation attachment is rarely important in NLP applications, and is not very likely to significantly contribute to cross-lingual dependency structure similarity either. For this reason, we also include UAS measured only on non-punctuation nodes. Still, adposition attachment, which we are mostly interested in, accounts for only a part of the score difference.

4.2 Conversion between Prague and Stanford adpositions

Further on, we only use the Prague style annotation of the treebanks, with adpositions annotated either in Prague style (P) or Stanford style (S). To convert between these adposition annotation styles, we implemented two simple transformation blocks in the Treex NLP framework (Žabokrtský, 2011):

- The conversion from P to S takes each adposition and attaches it as a dependent of its left-most non-adpositional child, as well as all of its other non-adpositional children. Thus, the adposition becomes a leaf node, unless it has adpositional dependent nodes (typically this signifies a compound adposition). Coordinating conjunctions are dived through – if the left-most non-adpositional child is a coordinating conjunction, the leftmost non-adpositional conjunct is taken instead (recursively).

- In the conversion from S to P, each adposition with a non-adpositional head is attached as a dependent of its head's head, and its original head is attached as its dependent.

The roundtrip of the conversion (UAS after converting there and back again) is around 98%. The transformation blocks,[7] as well as the whole Treex framework, are available on Github.[8]

Note that there are three places where a conversion from one annotation style to another may take place – conversion of the source treebank before training a parser, conversion of the parser output before the parse tree combination, and conversion of the parse tree combination output.

4.3 29-to-1 delexicalized parser transfer

This section presents and discusses annotation style conversions applied in semi-supervised parsing of each of the 30 HamleDT 2.0 treebanks as the *tgt*, using delex parsers trained on the remaining 29 *src* treebanks, in an unweighted parse tree combination approach.

As was already mentioned, one of the two adposition annotation styles (P or S) can be used for parsing, for parse tree combination, and for converting the output dependency tree. This yields a number of possible setups, which we will denote as the styles used for the individual steps, separated by slashes: parsing/combination/output.

For example, a "P/S/S" setup means we use the original P style of the treebanks for training the parsers, a conversion from P to S of the parse trees is performed before combining them, and no conversion of the resulting dependency trees takes place (as they already were in S style before the combination). In some of the experiments, we use both P and S parsing – "P,S/S/P" denotes a setup where both parsers trained on P and S treebanks were applied, the P style parse trees were then converted to S style, all of these were combined using the maximum spanning tree algorithm, and the output dependency tree was then converted to P style.

The results are shown in Table 3. For each target language, it shows the UAS of evaluating various setups on the test section of its treebank.

Clearly, the best results are obtained by parsing both to Stanford and Prague adposition annotation style, thus obtaining two parse trees for each source language (58 parse trees for each sentence), converting the parse trees to the desired output style, and combining them. This shows that both of the styles have some advantages, and that the parse tree combination can benefit from these. Moreover, contrary to supervised parsing, the S style performs better than the P style, by +0.26% UAS absolute on average. This is one of the indications that in this multi-lingual setting, the S style of adposition attachment is favourable, as it makes the dependency trees more similar. Overall, using both styles for parsing and Stanford style for combination and output surpasses the Prague-only baseline by +0.39% UAS absolute.

A further observation to make is that generally, the P style is good for parsing, while the S style is good for parse tree combination. Please note the results of P/P/P and S/P/P, which show a clear dominance (+1.17%) of using P style for everything rather than parsing in S and then converting to P. For the S output style, the difference between S/S/S and P/S/S is only 0.08% UAS, and parsing to P and then converting to S and combining actually achieves the best score for 8 target languages, while using S for everything leads to best results for only 7 languages. This can be easily explained,

[7] `HamleDT::Transform::PrepositionDownwardSimple` and `HamleDT::Transform::PrepositionUpwardSimple`

[8] `https://github.com/ufal/treex/` `https://github.com/ufal/treex/tree/master/lib/Treex/Block/HamleDT/Transform`

Tgt	P style output						S style output					
lang	P/P/P	P/S/P	S/S/P	S/P/P	P,S/P/P	P,S/S/P	S/S/S	S/P/S	P/P/S	P/S/S	S,P/S/S	S,P/P/S
ar	44.61	43.07	42.29	43.32	**44.99**	42.93	43.16	42.44	42.57	**44.39**	44.13	43.18
bg	**73.17**	71.13	70.91	71.65	72.72	71.27	72.24	72.51	**72.68**	72.55	72.65	72.58
bn	59.98	59.98	**60.47**	**60.47**	60.34	60.22	**60.47**	**60.47**	59.98	59.98	60.22	60.34
ca	**66.45**	64.78	64.49	65.32	66.38	64.73	65.61	66.08	66.10	**66.14**	66.07	66.12
cs	64.06	63.21	62.94	63.50	**64.14**	63.30	63.62	63.68	63.55	63.83	**63.93**	63.79
da	**63.74**	61.69	62.00	62.53	63.53	61.98	62.82	62.82	62.41	62.58	**63.09**	62.41
de	52.58	49.68	53.00	52.25	**55.17**	52.48	**55.95**	52.52	52.18	52.40	55.32	54.92
el	67.05	66.42	66.90	66.76	**67.69**	67.03	67.63	67.24	66.80	67.24	**67.78**	67.63
en	46.13	43.19	45.77	45.19	**48.23**	45.69	**47.65**	44.31	44.69	44.41	47.09	46.71
es	**69.73**	67.59	67.46	68.30	69.61	67.71	68.85	69.09	69.12	69.00	**69.17**	69.09
et	71.34	70.19	71.23	71.23	**72.07**	71.76	74.06	73.12	72.59	73.85	**74.48**	73.22
eu	46.12	**46.18**	46.15	45.97	45.92	46.07	46.15	45.97	46.12	**46.18**	46.07	45.92
fa	54.69	53.03	53.29	53.38	**54.77**	53.50	56.41	55.23	55.86	56.35	**56.69**	55.84
fi	**51.48**	51.34	50.47	50.40	51.17	50.99	50.60	50.54	**51.59**	51.47	51.08	51.34
grc	46.24	45.50	46.29	**46.55**	46.38	46.31	46.48	46.18	45.81	45.74	**46.50**	45.92
hi	30.12	**30.66**	28.96	28.41	29.64	29.60	33.23	30.81	32.10	**34.45**	33.64	31.25
hu	59.68	**60.20**	59.56	59.64	59.89	59.95	60.50	60.10	60.25	**61.02**	60.81	60.24
it	64.52	63.60	63.97	64.60	**65.13**	63.91	64.44	**64.72**	64.03	64.19	64.50	**64.72**
ja	**44.23**	40.94	39.36	40.27	42.64	39.87	44.02	41.55	44.02	**45.49**	44.88	42.81
la	41.14	40.72	41.16	40.80	**41.28**	41.22	41.34	40.93	41.28	41.05	**41.47**	**41.47**
nl	**62.47**	59.09	60.63	60.72	62.04	60.39	**63.81**	62.61	62.08	62.47	63.80	61.99
pt	71.35	69.78	69.97	70.97	**71.60**	70.09	71.14	**71.81**	70.94	71.02	71.26	71.76
ro	59.66	56.48	55.30	57.50	**59.85**	55.76	58.52	58.37	59.43	**59.51**	58.67	59.39
ru	**63.82**	63.36	62.20	62.78	63.65	62.84	62.43	62.87	**63.85**	63.68	63.13	63.48
sk	63.66	63.02	62.96	63.22	**63.73**	63.27	63.36	63.20	63.22	63.43	**63.62**	63.34
sl	**54.40**	53.35	53.16	53.24	53.68	53.15	53.80	53.18	**54.07**	53.94	53.68	53.27
sv	62.08	59.00	59.97	60.87	**62.18**	59.53	**62.22**	61.42	60.80	60.91	61.60	61.32
ta	38.76	38.51	36.70	37.76	**39.01**	38.21	37.66	37.91	38.86	**39.06**	38.91	**39.06**
te	66.83	66.83	**67.00**	66.83	66.16	66.50	**67.00**	66.83	66.83	66.83	66.50	66.16
tr	40.39	40.35	40.77	40.66	40.82	**40.93**	**41.28**	40.86	40.53	40.73	41.26	40.93
Avg	56.68	55.43	55.51	55.84	**56.81**	55.71	56.88	56.31	56.48	56.80	**57.07**	56.67

Table 3: UAS of various setups of delexicalized parser transfer, always using 1 language as target (*Tgt lang*) and the remaining 29 languages as source. The best result on each line for both of the output styles is highlighted in bold. The columns correspond to various combinations of annotation styles used for parsing/combining/output – P corresponds to Prague style and S to Stanford style adposition annotation.

Subset	ADP freq.	Language
High	15%	Spanish
	19%	Hindi
	19%	Japanese
Med	9%	Czech
	8%	English
	9%	Swedish
Low	0%	Basque
	4%	Ancient Greek
	1%	Hungarian
Mix	15%	Spanish
	9%	Swedish
	1%	Hungarian

Table 4: Subsets of treebanks, selected according to their frequency of adposition tokens.

Setup	High	Med	Low	Mix	All9
P/P/P	40.53	52.00	44.53	41.03	54.98
P/S/P	39.87	50.81	41.28	38.55	52.75
S/S/P	39.39	50.41	41.17	39.22	52.86
S/P/P	39.68	50.86	42.15	39.36	53.79
P,S/P/P	**41.29**	**52.57**	**45.00**	**41.75**	**55.37**
P,S/S/P	40.70	51.57	43.29	39.73	53.49
S/S/S	41.36	51.64	43.69	41.95	54.85
S/P/S	40.43	51.49	42.97	40.66	53.88
P/P/S	40.69	51.68	44.15	40.82	54.38
P/S/S	41.87	51.91	44.64	41.55	54.76
S,P/S/S	**42.77**	**52.67**	**46.41**	**42.66**	**55.42**
S,P/P/S	41.60	52.50	44.80	41.82	54.96

Table 5: UAS of delexicalized parser transfer, averaged over 21 languages; some or all of the remaining 9 languages are used as source, according to the given subset name.

as it has been already shown that the P style is generally favourable for parsing; the S style then makes the parse trees more similar and thus easier to combine correctly.

And finally, there is a general tendency of simpler solutions to perform better – unless there is a strong benefit of switching styles for a given step, it is preferable to use a low number of conversions.

4.4 Smaller source treebank subsets

For a deeper insight and further confirmation of our conclusions, we also performed a set of experiments with smaller subsets of the treebank collection. We selected several treebank groups, based on the ratio of adposition tokens to all tokens. We also only chose large enough treebanks (more than 100,000 tokens). The subsets are listed in Table 4; we also used a larger “All9” set of all the 9 selected treebanks. Only these were then used for training; the remaining 21 languages were used for testing as target languages.

The summary results are to be found in Table 5. It is easy to see that the conclusions presented in the previous section hold even for these datasets. Moreover, the differences in UAS are often much higher, especially for the smaller and highly diverse datasets High, Low and Mix, where the benefit of the Stanford style making the dependency trees more similar becomes quite important. This suggests that the role of Stanford style is stronger with small and heterogeneous datasets. For the High dataset, the best result surpasses the Prague-only baseline by as much as 2.24% UAS absolute.

4.5 Supervised parsers

For completeness, we also include results for supervised monolingual lexicalized and delexicalized parsers, using the P and S annotation styles of adpositions. The setup denoted as “P/S” corresponds to a parser trained on a P style target treebank, output of which is converted to S, and then evaluated on the S conversion of the target treebank test section (analogously for “S/P”). For comparison, we also include the two best parser transfer setups (these results are identical to those in Table 3).

The results are shown in Table 6. For the lexicalized parser, the P style is clearly better, achieving +0.77% UAS absolute on average. To obtain S style parse trees, it is generally better to parse the text using a parser trained on a P style treebank, and then to convert the output parse trees, which yields a +0.46% higher UAS than parsing directly using an S style treebank. Here, the adpositions clearly provide important information to the parser, and their annotation as heads benefits the results.

For the delexicalized parser, the P style still performs better (+0.21% UAS), although the difference is smaller, and parsing directly using the S style is comparable to parsing using P style and then converting to S style. We believe that this is because when word forms and lemmas are removed, the most important information about the adpositions is missing. If the language has a general tendency of where it attaches adpositions, the information that a word is an adposition is still useful, but it has now a limited power towards adposition attachment disambiguation.

And finally, as has already been discussed, the S style actually performs better for delexicalized parser transfer than the P style; in the best setups, the S style achieves +0.26% UAS on average.

Tgt	Lexicalized supervised				Delexicalized supervised				Transfer	
lang	P	S/P	S	P/S	P	S/P	S	P/S	P,S/P/P	S,P/S/S
ar	**77.47**	73.92	76.32	77.17	**69.61**	67.24	69.29	69.50	**44.99**	44.13
bg	**87.95**	85.83	87.50	87.61	**83.87**	81.11	82.76	83.32	**72.72**	72.65
bn	82.27	**82.39**	**82.39**	82.27	77.59	**78.82**	**78.82**	77.59	**60.34**	60.22
ca	**86.11**	81.51	84.37	85.49	**79.71**	76.41	79.03	79.33	**66.38**	66.07
cs	**80.87**	79.43	80.31	80.63	**70.99**	70.06	70.69	70.69	**64.14**	63.93
da	**85.66**	82.55	84.42	85.12	**81.13**	78.37	80.31	80.67	**63.53**	63.09
de	**84.65**	82.24	83.57	84.53	**77.52**	75.55	76.92	77.47	55.17	**55.32**
el	**80.68**	79.41	80.20	80.18	**75.40**	74.10	75.15	74.73	67.69	**67.78**
en	**84.71**	82.85	84.37	84.05	**76.57**	74.92	76.19	76.03	**48.23**	47.09
es	**85.46**	80.41	83.55	84.74	**79.75**	75.57	78.52	79.25	**69.61**	69.17
et	85.15	85.25	**86.30**	85.46	80.96	81.38	**82.85**	80.75	72.07	**74.48**
eu	**75.28**	75.07	75.07	**75.28**	68.34	**68.41**	**68.41**	68.34	45.92	**46.07**
fa	**82.27**	77.94	80.21	81.70	70.44	69.17	**71.72**	70.78	54.77	**56.69**
fi	71.17	70.62	70.80	**71.21**	63.10	62.25	62.51	**63.13**	**51.17**	51.08
grc	**56.98**	56.41	56.61	56.56	48.92	48.80	**49.10**	48.80	46.38	**46.50**
hi	90.40	83.92	86.43	**90.42**	**80.55**	78.15	80.52	80.52	29.64	**33.64**
hu	**77.60**	77.03	77.07	77.40	**72.54**	71.80	71.79	72.34	59.89	**60.81**
it	**81.46**	80.06	80.57	81.22	**77.49**	76.26	76.57	76.92	**65.13**	64.50
ja	**91.17**	84.54	89.65	90.79	81.72	79.83	84.03	**84.35**	42.64	**44.88**
la	47.55	48.69	**48.72**	47.36	44.08	43.89	**44.12**	43.81	41.28	**41.47**
nl	**80.90**	77.37	80.05	80.11	**74.02**	70.98	73.70	73.57	62.04	**63.80**
pt	**83.50**	80.62	82.21	82.97	**80.14**	76.99	78.68	79.77	**71.60**	71.26
ro	**89.62**	86.52	88.79	**89.62**	85.19	83.41	**85.34**	84.85	**59.85**	58.67
ru	**83.98**	82.91	83.49	83.75	**73.08**	72.24	72.70	72.90	**63.65**	63.13
sk	**79.02**	78.19	78.70	78.63	**71.38**	70.60	70.88	70.93	**63.73**	63.62
sl	**81.19**	80.05	80.94	80.95	72.91	72.30	**72.93**	72.69	**53.68**	**53.68**
sv	**83.20**	79.31	81.93	82.48	**78.84**	75.67	77.97	78.18	**62.18**	61.60
ta	72.70	**72.75**	72.60	72.30	**68.17**	67.82	67.92	67.62	**39.01**	38.91
te	**87.60**	86.60	86.93	**87.60**	**85.59**	83.75	84.09	85.59	66.16	**66.50**
tr	**79.48**	78.77	79.02	79.26	**73.99**	73.74	73.72	73.72	40.82	**41.26**
Avg	**80.54**	78.44	79.77	80.23	**74.12**	72.65	73.91	73.94	56.81	**57.07**

Table 6: UAS of supervised lexicalized and delexicalized monolingual parsers, as well as the two best-performing transfer parser setups. For the supervised parsers, the columns corresponds to annotation style used for parsing, or parsing/output if a conversion was performed after parsing. For the delexicalized parser transfer, the columns correspond to parsing with both P and S style, converting the parse trees to the same style (P or S), and combining the trees.

4.6 Conclusion

We investigated the usefulness of Stanford adposition attachment style as an alternative to the Prague style, using a large set of 30 treebanks for evaluation. We especially focused on multi-source cross-lingual delexicalized parser transfer, as one of the targets behind the design of Universal Stanford Dependencies is to be more cross-lingually consistent than other annotation styles.

We managed to confirm that for supervised parsing, Prague annotation style is favourable over Stanford style, as has been already stated in literature. However, in the parser transfer setting, Stanford style adposition attachment proved to generally perform better than the Prague style, thanks to its abstraction from the high interlingual variance in adposition usage. Moreover, even better results are achieved by at once combining outputs of parsers trained on treebanks of both Prague and Stanford adposition attachment style, eventually reaching an improvement of +0.39% UAS absolute over the Prague style baseline. Our results are further confirmed by experiments using smaller and more diverse subsets of training treebanks, where the advantage of Stanford style often becomes even more pronounced, reaching an improvement of up to +2.24% over the Prague style baseline.

In future, we intend to evaluate the effect of other annotation style differences, such as the coordination structures. We also plan to try to incorporate more fine-grained morphological information than the UPOS tags, probably by only including a given feature if it seems to be shared between the src and tgt language, as always including all of them performed very poorly in preliminary experiments.

5 Employing Language Similarity

The issue of finding a good src treebank for a given tgt language can be approached in two ways. In the single-source approach, we try to find the src language which is most similar to the tgt language, and use the treebank for that language to train a parser to be applied to the tgt. In a multi-source approach, we combine some or all available src resources, either in an unweighted way, as was presented in Section 3, or weighted by src-tgt similarity. Thus, for both the source selection task and the source weighting task, there is a need for a language similarity measure, which serves as a proxy for src and tgt treebank similarity, but cannot access the tgt treebank. Still, it is reasonable, and usual, to presuppose availability of POS-tagged tgt language text (as it constitutes the input to the delex parser), as well as the information about the identity of the tgt language.

Several authors (Naseem et al., 2012; Søgaard and Wulff, 2012; Täckström et al., 2013) have employed the World Atlas of Language Structures (WALS) of Dryer and Haspelmath (2013) to estimate the similarity of languages for delex transfer. They exploit information about the genealogy distance and shared typological features of the languages, typically word order features. They note that for a tgt language which is rather dissimilar to any of the src languages, delex transfer achieves better results if word order is completely or selectively ignored. This is motivated by the observation that languages, at least when being observed only through POS, become more similar if we disregard word order.

Apart from using WALS, Søgaard also takes one other approach to estimating language similarity. In (Søgaard, 2011), he trains a POS language model on a tgt POS-tagged corpus, and uses it to filter the src treebank, keeping only sentences that look like target language sentences to the language model. This method is further improved in (Søgaard and Wulff, 2012), where the authors move from a selection approach to a weighting approach: they keep all the src sentences, but weight each of them with the score assigned by the language model. This is made possible by modifying their learning algorithm to use weighted perceptron learning (Cavallanti et al., 2010). In this way, they in principle introduce a weighting scheme for the treebank concatenation.

The central feature of this section is KL_{cpos^3}, a language similarity measure based on similarity of distributions of coarse POS tag trigrams, computed over POS-tagged corpora for the source and target languages using the Kullback-Leibler divergence (Kullback and Leibler, 1951). The measure is simple and efficient, does not rely on additional external resources, and has been designed and tuned specifically to be used in delexicalized transfer approaches. We show that KL_{cpos^3} performs well in selecting the source treebank in the single-source delexicalized transfer, as well as in parser weighting in the multi-source tree combination approach.

5.1 KL_{cpos^3} language similarity measure

Our method of estimating language similarity for the purposes of delexicalized transfer is based on comparing distributions of coarse POS trigrams in a source language treebank (P_{src}) and in a target language POS-tagged corpus (P_{tgt}). This is motivated by the fact that POS tags constitute a key feature for delexicalized parsing. We use UPOS tags; we also tried using more fine-grained tags, but this led to worse results on our development data, probably because more fine-grained features tend to be less shared across languages. We also tried to vary the POS $n$gram length; bigrams and tetragrams both performed comparably to trigrams on the weighting task, but for the selection task, trigrams outperformed other $n$grams.

The coarse POS trigram distributions are estimated as frequencies computed on the training parts of the corpora:

$$f(cpos_{i-1}, cpos_i, cpos_{i+1}) = \\ = \frac{\text{count}(cpos_{i-1}, cpos_i, cpos_{i+1})}{\sum_{\forall cpos_{a,b,c}} \text{count}(cpos_a, cpos_b, cpos_c)}\,; \quad (2)$$

we use a special value for $cpos_{i-1}$ or $cpos_{i+1}$ if $cpos_i$ appears at the beginning or end of a sentence, respectively.

We use the Kullback-Leibler divergence[9] to compute the similarity of the distributions as $D_{\text{KL}}(P_{tgt}||P_{src})$.[10] The KL divergence of distributions P and Q is defined as

$$D_{\text{KL}}(P||Q) = \sum_{\forall x} P(x) \cdot \log \frac{P(x)}{Q(x)}\,, \quad (3)$$

with the value of the addend defined as 0 if $P(x) = 0$. The value of KL divergence is a nonnegative number; the more divergent (dissimilar) the distributions, the higher its value.

In our setting, we compute KL_{cpos^3} as

$$KL_{cpos^3}(tgt, src) = \\ = \sum_{\forall cpos^3 \in tgt} f_{tgt}(cpos^3) \cdot \log \frac{f_{tgt}(cpos^3)}{f_{src}(cpos^3)}\,, \quad (4)$$

where $cpos^3$ is a coarse POS tag trigram. It is sufficient to iterate only over trigrams present in the target, as the addend is defined to be zero in other cases. This is in accord with our needs: we do not actually care about phenomena that the source parser can handle but do not appear in target.

For the KL divergence to be well-defined, we set the source count of each unseen trigram to 1.

5.2 Source selection

For the single-source parser transfer, we compute KL_{cpos^3} distance of the tgt corpus to each of the src treebanks. We then select the src^* treebank as the closest one:

$$src^* = \underset{\forall src}{\arg\min}\; KL_{cpos^3}(tgt, src)\,, \quad (5)$$

and use it to train the delex parser to be applied to tgt.

5.3 Source weighting

To convert KL_{cpos^3} from a negative measure of language similarity to a positive *src* parser weight, we take the fourth power of its inverted value, $KL_{cpos^3}^{-4}$. A high value of the exponent strongly promotes the most similar source language, giving minimal power to the other languages, which is good if there is a *very* similar source language. A low value enables combining information from a larger number of source languages. We chose a compromise value of 4 based on performance on the development data.

We then add weighting by KL_{cpos^3} into the parse tree combination (Section 3.3) by multiplying the contribution of each *src* parse tree by $KL_{cpos^3}^{-4}$ (*tgt*, *src*).

5.4 Evaluation

To evaluate the language similarity measure, we use it both for the selection task and the weighting task on the stanfordized version of the HamleDT 2.0 treebanks. The exact shape of the measure was tuned on the development set.[11]

Preliminary trials on the subset of CoNLL 2006 and 2007 data sets (Buchholz and Marsi, 2006; Nilsson et al., 2007) used by McDonald et al. (2011) indicated that these are not suitable for our approach, as they are not harmonized on the dependency annotation level.[12] There, treebank annotation style similarity seems to become more

[9] We also tried cosine similarity, with much worse results.

[10] $D_{\text{KL}}(P||Q)$ expresses the amount of information lost when a distribution Q is used to approximate the true distribution P. Thus, in our setting, we use $D_{\text{KL}}(P_{tgt}||P_{src})$, as we try to minimize the loss of using a parser based on source data as an approximation of a parser based on the target data.

[11] We tuned the choice of the similarity measure, POS n-gram length, and the way of turning KL_{cpos^3} into $KL_{cpos^3}^{-4}$.

[12] On the original non-harmonized treebanks, the unweighted tree combination performed best (58.06% UAS), +2.4% absolute over weighted tree combination and +4.9%

important than language similarity; the lack of harmonization makes the data unnecessarily noisier.

Table 7 contains the results of our methods both on the test languages and the development languages. For each target language, we used all remaining 29 source languages for training (in the single-source method, only one of them is selected and applied). We base our evaluation mainly on average UAS on the test tgt languages, and compare the methods by absolute UAS differences.

Our baseline is the treebank concatenation method of McDonald et al. (2011), i.e., a single delexicalized parser trained on the concatenation of the 29 src treebanks (Section 3.2).

As an upper bound,[13] we report the results of the oracle single-source delexicalized transfer: for each target language, the oracle source parser is the one that achieves the highest UAS on the target treebank test section.[14] We do not include results of a higher upper bound of a supervised delexicalized parser (trained on the tgt treebank), which has an average UAS of 68.5%. It was not surpassed by our methods for any target language, although it was reached for Telugu, and approached within 5% for Czech and Latin.

The results show that KL_{cpos^3} performs well both in the selection task and in the weighting task, as both the single-source and the weighted multi-source transfer methods outperform the unweighted tree combination on average, as well as the treebank concatenation baseline. In 8 of 18 cases, KL_{cpos^3} is able to correctly identify the oracle source treebank for the single-source approach. In two of these cases, weighted tree combination further improves upon the result of the single-source transfer, i.e., surpasses the oracle; in the remaining 6 cases, it performs identically to the single-source method. This proves KL_{cpos^3} to be a successful language similarity measure for delexicalized parser transfer, and the weighted multi-source transfer to be a better performing approach than the single-source transfer.

The weighted tree combination is better than its unweighted variant only for half of the target languages, but it is more stable, as indicated by its

Tgt	TB	Oracle		Single-src			Multi-src	
lang	conc	del trans		KL			×1	×w
bn	61.0	te	**66.7**	0.5	**te**	**66.7**	63.2	**66.7**
cs	60.5	sk	**65.8**	0.3	**sk**	**65.8**	60.4	**65.8**
da	**56.2**	en	**55.4**	0.5	sl	42.1	**54.4**	50.3
de	12.6	en	**56.8**	0.7	**en**	**56.8**	27.6	**56.8**
en	12.3	de	**42.6**	0.8	**de**	**42.6**	21.1	**42.6**
eu	**41.2**	da	**42.1**	0.7	tr	29.1	**40.8**	30.6
grc	43.2	et	42.2	1.0	sl	34.0	**44.7**	42.6
la	38.1	grc	40.3	1.2	cs	35.0	**40.3**	39.7
nl	55.0	da	57.9	0.7	**da**	57.9	56.2	**58.7**
pt	62.8	en	64.2	0.2	es	62.7	**67.2**	62.7
ro	44.2	it	**66.4**	1.6	la	30.8	**51.2**	50.0
ru	55.5	sk	57.7	0.9	la	40.4	**57.8**	57.2
sk	52.2	cs	**61.7**	0.2	sl	58.4	**59.6**	58.4
sl	45.9	sk	**53.9**	0.2	**sk**	**53.9**	47.1	**53.9**
sv	45.4	de	**61.6**	0.6	da	49.8	**52.3**	50.8
ta	27.9	hi	**53.5**	1.1	tr	31.1	28.0	**40.0**
te	67.8	bn	**77.4**	0.4	**bn**	**77.4**	68.7	**77.4**
tr	18.8	ta	40.3	0.7	**ta**	40.3	23.2	**41.1**
Avg	44.5		**55.9**	0.7		48.6	48.0	**52.5**
SD	16.9		10.8			14.4	15.0	11.8
ar	37.0	ro	**43.1**	1.7	sk	41.2	35.3	**41.3**
bg	64.4	sk	66.8	0.4	**sk**	66.8	66.0	**67.4**
ca	56.3	es	**72.4**	0.1	**es**	**72.4**	61.5	**72.4**
el	63.1	sk	61.4	0.7	cs	60.7	62.3	**63.8**
es	59.9	ca	**72.7**	0.0	**ca**	**72.7**	64.3	**72.7**
et	67.5	hu	71.8	0.9	da	64.9	70.5	**72.0**
fa	30.9	ar	**35.6**	1.1	cs	**34.7**	32.5	33.3
fi	41.9	et	44.2	1.1	**et**	44.2	41.7	**47.1**
hi	24.1	ta	**56.3**	1.1	fa	20.8	24.6	**27.2**
hu	55.1	et	52.0	0.7	cs	46.0	**56.5**	51.2
it	52.5	ca	**59.8**	0.3	pt	54.9	59.5	**59.6**
ja	29.2	tr	**49.2**	2.2	ta	**44.9**	28.8	34.1
Avg	48.5		**57.1**	0.9		52.0	50.3	**53.5**
SD	15.2		12.5			16.1	16.5	16.7
Avg	46.1		**56.4**	0.8		50.0	48.9	**52.9**
SD	16.1		11.3			15.0	15.4	13.7

Table 7: Evaluation using UAS on test target treebanks (upper part of the table) and development target treebanks (lower part).

For each target language, all 29 remaining non-target treebanks were used for training the parsers. The best score among our transfer methods is marked in bold; the baseline and upper bound scores are marked in bold if equal to or higher than that.

Legend:
Tgt lang = Target treebank language.
TB conc = Treebank concatenation.
Oracle del trans = Single-source delexicalized transfer using the oracle source language.
Single-src = Single-source delexicalized transfer using source language with lowest KL_{cpos^3} distance to the target language (language bold if identical to oracle).
Multi-src = Multi-source delexicalized transfer using parse tree combination, unweighted ($\times 1$) and $KL_{cpos^3}^{-4}$ weighted ($\times w$).
Avg = Average UAS (on test/development/all).
SD = Standard sample deviation of UAS, serving as an indication of robustness of the method.

over single-source transfer. On the harmonized versions of the same treebanks (subset of HamleDT), unweighted tree combination was outperformed both by single-source transfer (+1.0%) and weighted tree combination (+1.67%).

[13] This is a hard upper-bound for the single-source transfer, but can be surpassed by the multi-source transfer.

[14] We do not report the matrix of all source/target combination results, as this amounts to 870 numbers.

lower standard deviation, and achieves an average UAS higher by 4.5%. The unweighted tree combination, as well as treebank concatenation, perform especially poorly for English, German, Tamil, and Turkish, which are rich in determiners, unlike the rest of the treebanks;[15] therefore, determiners are parsed rather randomly.[16] In the weighted methods, this is not the case anymore, as for a determiner-rich target language, determiner-rich source languages are given a high weight.

For target languages for which KL_{cpos^3} of the closest source language was lower or equal to its average value of 0.7, the oracle treebank was identified in 7 cases out of 12 and a different but competitive one in 2 cases; when higher than 0.7, an appropriate treebank was only chosen in 1 case out of 6. When KL_{cpos^3} failed to identify the oracle, weighted tree combination was always better or equal to single-source transfer but mostly worse than unweighted tree combination. This shows that for distant languages, KL_{cpos^3} does not perform as good as for close languages.

We believe that taking multiple characteristics of the languages into account would improve the results on distant languages. A good approach might be to use an empirical measure, such as KL_{cpos^3}, combined with supervised information from other sources, such as WALS. Alternatively, a backoff approach, i.e. combining KL_{cpos^3} with e.g. KL_{cpos^2}, might help to tackle the issue.

Still, for target languages dissimilar to any source language, a better similarity measure will not help much, as even the oracle results are usually poor. More fine-grained resource combination methods are probably needed there, such as selectively ignoring word order, or using different sets of weights based on POS of the dependent node.

5.5 Conclusion

We presented KL_{cpos^3}, an efficient language similarity measure designed for delexicalized dependency parser transfer. We evaluated it on a large set of treebanks, and showed that it performs well in selecting the source treebank for single-source transfer, as well as in weighting the source treebanks in multi-source parse tree combination.

Our method achieves good results when applied to similar languages, but its performance drops for distant languages. In future, we intend to explore combinations of KL_{cpos^3} with other language similarity measures, so that similarity of distant languages is estimated more reliably.

6 Model Interpolation

In this section, we present a novel method for src information combination, based on interpolation of trained MSTperl parser models. Our approach was motivated by an intuition that the more fine-grained information provided by the src edge scores could be of benefit, probably serving as src parser confidence. Moreover, model interpolation is significantly less computationally demanding at inference than the parse tree combination method, as instead of running a set of separate src parsers, only one parser is run.

We are not aware of any prior work on interpolating dependency parser models; the closest to our approach is the interpolation of multilingual probabilistic context-free grammars of Cohen et al. (2011).

The method proceeds as follows:

1. Train a delex parser model on each src treebank (Section 3.1).
2. Normalize the parser models (Section 6.1).
3. Interpolate the parser models (Section 6.2, Section 6.3).
4. Parse the tgt text with a delex parser using the interpolated model.

We evaluate the model interpolation method in Section 6.4, comparing it both to the treebank concatenation method as well as the parse tree combination method, in a weighted as well as unweighted setting.

6.1 Model normalization

An important preliminary step to model interpolation is to normalize each of the trained models, as the feature weights in models trained over different treebanks are often not on the same scale (we do not perform any regularization during the parser training). We use a simplified version of normalization by standard deviation. First, we compute the uncorrected sample standard deviation of the weights of the features in the model as

$$s_M = \sqrt{\frac{1}{|M|} \sum_{\forall f \in M} (w_f - \bar{w})^2}\,, \qquad (6)$$

[15] In the treebanks for these four languages, determiners constitute around 5-10% of all tokens, while most other treebanks contain no determiners at all; in some cases, this is related to properties of the treebank annotation or its harmonization rather than properties of the language.

[16] UAS of determiner attachment tends to be lower than 5%, which is several times less than for any other POS.

where $\bar{w}$ is the average feature weight, and $|M|$ is the number of feature weights in model M; only features that were assigned a weight by the training algorithm are taken into account.

We then divide each feature weight by the standard deviation:[17]

$$\forall f \in M : w_f := \frac{w_f}{s_M} \,. \tag{7}$$

The choice of normalization by standard deviation is a combination of its high and stable performance on our development set, and of Occam's razor. We tried 12 normalization schemes, nearly all of which achieved an improvement of 2.5% to 5% UAS absolute over an interpolation of unnormalized models on average, but often with large differences for individual languages.[18]

6.2 Unweighted model interpolation

The interpolated model is a linear combination of the normalized models trained over the *src* treebanks. The result is a model that can be used in the same way as a standard MSTperl parser model.

In unweighted model interpolation, the weight of each feature (w_f) is computed as the sum of the weights of that feature in the normalized *src* models ($w_{f,src}$):

$$\forall f \in F : w_f = \sum_{\forall src} w_{f,src} \,. \tag{8}$$

6.3 Weighted model interpolation

In the weighted variant of model interpolation, we extend (8) with multiplication by a weight $w(tgt, src)$, corresponding to language similarity of tgt and src:

$$\forall f \in F : w_f = \sum_{\forall src} w_{f,src} \cdot w(tgt, src) \,. \tag{9}$$

In our experiments, we use the $KL^{-4}_{cpos^3}(tgt, src)$ weight, which we presented in Section 5.

6.4 Evaluation

Table 8 contains the results of our model interpolation methods, as well as the baseline methods. For

Target language	Unweighted Conc	Unweighted Tree	Unweighted Inter	Weighted Tree	Weighted Inter
Bengali	61.0	63.2	**67.1**	66.7	**66.9**
Czech	**60.5**	60.4	57.5	**65.8**	65.2
Danish	**56.2**	54.4	48.9	**50.3**	49.5
German	12.6	**27.6**	18.2	56.8	**61.6**
English	12.3	**21.1**	16.2	42.6	**48.6**
Basque	**41.2**	40.8	39.5	30.6	**34.9**
Anc. Greek	43.2	**44.7**	41.4	42.6	**44.0**
Latin	38.1	**40.3**	39.7	**39.7**	39.5
Dutch	55.0	**56.2**	54.2	58.7	**59.4**
Portuguese	62.8	**67.2**	62.8	62.7	**63.7**
Romanian	44.2	**51.2**	48.6	50.0	**50.3**
Russian	55.5	**57.8**	53.3	**57.2**	56.3
Slovak	52.2	**59.6**	55.7	58.4	**60.6**
Slovenian	45.9	**47.1**	42.8	**53.9**	49.6
Swedish	45.4	**52.3**	49.4	**50.8**	50.4
Tamil	27.9	**28.0**	27.6	**40.0**	37.3
Telugu	67.8	68.7	**72.9**	**77.4**	**77.4**
Turkish	18.8	23.2	**25.3**	**41.1**	34.8
Average	44.5	**48.0**	45.6	52.5	**52.8**
Std. dev.	16.9	**15.0**	16.0	**11.8**	12.0
Arabic	**37.0**	35.3	30.7	**41.3**	34.6
Bulgarian	64.4	**66.0**	60.3	67.4	**68.5**
Catalan	56.3	**61.5**	58.5	72.4	**72.4**
Greek	**63.1**	62.3	59.6	63.8	**64.1**
Spanish	59.9	**64.3**	60.4	72.7	**72.7**
Estonian	67.5	**70.5**	67.4	**72.0**	71.7
Persian	30.9	**32.5**	29.5	**33.3**	28.6
Finnish	**41.9**	41.7	41.5	**47.1**	44.7
Hindi	24.1	24.6	**26.2**	27.2	**32.7**
Hungarian	55.1	56.5	**57.4**	51.2	**53.0**
Italian	52.5	**59.5**	56.0	59.6	**60.1**
Japanese	**29.2**	28.8	27.2	**34.1**	33.0
Average	48.5	**50.3**	47.9	**53.5**	53.0
Std. dev.	**15.2**	16.5	15.6	**16.7**	17.4
Average	46.1	**48.9**	46.5	**52.9**	52.9
Std. dev.	16.1	**15.4**	15.6	**13.7**	14.1

Table 8: UAS on test *tgt* treebanks (upper part of table) and development *tgt* treebanks (lower part).
Conc = Treebank concatenation
Tree = Parse tree combination
Inter = Model interpolation
Average = Average UAS (on test/development/all)
Std. dev. = Standard sample deviation of UAS, serving as an indication of robustness of the method

[17] We have not found any further gains in performance when subtracting the sample mean from the weight before the division; the MSTParser models seem to be typically centered very similarly.

[18] Another well-performing method was to divide each feature weight by the sum of absolute values of all feature weights in the model; or a similar method, applied during inference individually for each sentence, using only the feature weights that fired for the sentence to compute the divisor.

each *tgt* language, all remaining 29 *src* treebanks were used for parser training. We base our evaluation on comparing absolute differences in UAS on the whole set of 30 languages as targets.

The performance of the weighted model interpolation is comparable to the weighted tree combination – the difference in average UAS of the methods is lower than 0.1%, with model interpolation achieving a higher UAS than the tree combination for 16 of the 30 *tgt* languages. This shows that weighted model interpolation is a good alternative to weighted tree combination.

In the unweighted setting, the situation is quite different, with model interpolation scoring much lower than tree combination (-2.4%), and only slightly higher than treebank concatenation (+0.4%) on average. This suggests that, contrary to our original intuition, edge scores assigned by the *src* models are not a good proxy for parser confidence, not even when appropriately normalized.[19] Furthermore, the weighted methods generally outperform the unweighted ones (by +4.0% for tree combination and by +6.4% for model interpolation on average), which suggests, among other, that the *src*-*tgt* language similarity is much more important than the exact values of *src* edge scores for resource combination in delex transfer.

6.5 Conclusion

We presented trained parser model interpolation as an alternative method for multi-source crosslingual delexicalized dependency parser transfer. Evaluation on a large collection of treebanks showed that in a setting where the source languages are weighted by their similarity to the target language, model interpolation performs comparably to the parse tree combination approach. Moreover, model interpolation is significantly less computationally demanding than the tree combination when parsing the target text, as the interpolation can be efficiently performed beforehand, thus only requiring to invoke a single parser at runtime, while in the tree combination approach, each source parser has to be invoked individually.

In the unweighted setting, model interpolation consistently performed much worse than tree combination, which we find rather surprising, and we therefore plan to further investigate this in future. Still, the weighted methods generally outperformed the unweighted ones, and as the language similarity measure that we used only requires the source treebanks and a target POS-tagged text, i.e. exactly the resources that are required even for the unweighted delex transfer methods, there is little reason not to employ the weighting. Therefore, the low performance of the unweighted model interpolation is of less importance than its high performance in the weighted setting.

7 Cross-lingual Lexicalization

A very popular method of improving the results of delexicalized parser transfer is by lexicalizing the parser in a semi-supervised manner (as manually created parallel treebanks are extremely rare). We did not explore that approach in this work; instead, we focused on improving the underlying delexicalized parser transfer. However, we plan to combine it with semi-supervised lexicalization in future (preliminary experiments indicate that this leads to further improvements).

7.1 Employing parallel data

The typical approach to lexicalization in delex parsing is by using dictionaries, parallel texts, and/or machine translation techniques (Zhao et al., 2009; McDonald et al., 2011; Täckström et al., 2012; Durrett et al., 2012; Ramasamy et al., 2014).

One option is to translate the *tgt* sentence into the *src* language, which then makes it possible to use a lexicalized *src* parser instead of a delexicalized one. The correspondence of the words in the translation to the *src* words can be established by using word alignment.

If one-to-many or many-to-many alignment is used, the projection of the syntactic structure through the alignment is non-trivial. Therefore, word-to-word translation can be used instead. Another option is to only include the *src* information through additional features, as has been done by Rosa et al. (2012), which does not require a one-to-one alignment.

If high-quality *src*-*tgt* parallel texts are available (i.e. created by human translators, not machine translation systems), they may be used instead to create an automatic parallel treebank, without the need to use machine translation (but using a word aligner is still necessary).

A major drawback of all of these methods is the fact that in most cases, large parallel texts, and thus high-quality machine translation, is only

[19] The same tendency was observed across all normalization methods evaluated on the development set.

available for English as the *src* language. For some *tgt* languages, there may be other well-resourced *src* languages, but generally, and especially for the focus languages, i.e. under-resourced ones, we are constrained to only using English *src*. As our experiments showed, the English treebank is rarely a good *src* treebank for delex parser transfer; therefore, we expect that the English-based lexicalization can only serve as a complement to the methods described in this paper, still using all available *src* treebanks.

7.2 Employing word embeddings

Recently, especially since the introduction of the word2vec tool by Mikolov et al. (2013), continuous vector space word representations, also known as word embeddings, have gained huge popularity, and have proven to be useful in many tasks of natural language processing.

In our setting, we are especially interested in the approaches that compute bilingual word vectors, trained to assign similar vectors to words with similar meaning, regardless of whether these are *src* or *tgt* language words. In this way, we could replace the lexical features by embedding features, thus circumventing the lexicalization problem.

It has to be noted though that our parser, as well as parsers of other authors, generally support only categorial features; at least in combinations, but non-combined features are of extremely limited usefulness. Thus, it is necessary to either convert the vectors from the continuous space to categorial features, thus losing many of their attractive properties, or to use a parser that naturally supports combinations of continuous features, which to the best of our knowledge is not available in present, although we are aware of ongoing research in this field.

7.3 Self-training

Some lexicalization can also be achieved in an unsupervised way by applying self-training (McClosky et al., 2006), i.e. parsing a (preferably large) POS-tagged *tgt* corpus by the delex transfer method, and then using the resulting automatic treebank to train a standard lexicalized parser in a supervised way. While it may seem that such a parser has no chance of outperforming the delex parser, this is not entirely true, as simply the presence or non-presence of some phenomena in the corpus may help to adjust some parameters of the parser. Moreover, for practical reasons, it may be useful to obtain a standard *tgt* parser model to apply for analyzing new *tgt* data, rather than always applying the full multisource transfer machinery.

Our preliminary experiments performed using the training sections of the *tgt* treebanks indicate a small but consistent improvement brought by this approach, both when training a lexicalized as well as a delexicalized parser on the automatically parsed data.

The self-training method is quite orthogonal to all the other approaches, and can thus presumably be applied on top of any future parsing system.

8 Conclusion

We presented our work on multi-source crosslingual transfer of delexicalized dependency parsers.

We evaluated the influence of treebank annotation styles on parsing performance, focusing on adposition attachment style, and found that the Stanford annotation, while being infavourable for supervised parsers, performs promisingly in the multilingual delexicalized parser transfer setting.

We then presented KL_{cpos^3}, an empirical language similarity measure designed for source parser weighting in multi-source delexicalized parser transfer. We demonstrated its generally good performance, although improvements still have to be made for cases where the target language is too dissimilar to any available source language.

And finally, we introduced a novel resource combination method, based on interpolation of trained MSTParser models. Although we found its performance to be below our expectations, when combined with $KL_{cpos^3}^{-4}$ weighting, its results match these of the weighted parse tree combination method. As model interpolation is less computationally demanding than parse tree combination, we find it to be a good alternative multi-source delexicalized parser transfer method.

Throughout our work, we also identified numerous promising paths for further research, the most important being semi-supervised lexicalization of the methods.

Acknowledgments

This research was supported by the grants GAUK 1572314 and SVV 260 224. This work has been using language resources developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2010013).

References

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1455–1465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. 2010. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 50–61, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Natalia Silveira, Timothy Dozat, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proc. of LREC'14*, Reykjavík, Iceland. European Language Resources Association (ELRA).

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11. Association for Computational Linguistics.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Angelina Ivanova, Stephan Oepen, and Lilja Øvrelid. 2013. Survey on parsing three dependency representations for english. In *ACL (Student Research Workshop)*, pages 31–37.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. 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 152–159, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 62–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*.

Joakim Nivre, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajič, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.0.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC-2012*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Martin Popel, David Mareček, Jan Štěpánek, Daniel Zeman, and Zdeněk Žabokrtský. 2013. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 517–527, Sofija, Bulgaria. Bălgarska akademija na naukite, Association for Computational Linguistics.

Loganathan Ramasamy, David Mareček, and Zdeněk Žabokrtský. 2014. Multilingual dependency parsing: Using machine translated texts instead of parallel corpora. *The Prague Bulletin of Mathematical Linguistics*, 102:93–104.

Rudolf Rosa, Ondřej Dušek, David Mareček, and Martin Popel. 2012. Using parallel features in parsing of machine-translated sentences for correction of grammatical errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea. ACL.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. 2014. HamleDT 2.0: Thirty dependency treebanks stanfordized. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 2334–2341, Reykjavík, Iceland. European Language Resources Association.

Rudolf Rosa. 2014. MSTperl parser.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132. Association for Computational Linguistics.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *Proceedings of COLING 2012: Technical Papers*.

Petr Sgall. 1967. Functional sentence perspective in a generative description. *Prague studies in mathematical linguistics*, 2(203-225).

Anders Søgaard and Julie Wulff. 2012. An empirical etudy of non-lexical extensions to delexicalized transfer. In *COLING (Posters)*, pages 1181–1190.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Anders Søgaard. 2013. An empirical study of differences between conversion schemes and annotation guidelines. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013), Prague, Czech Republic: Charles University in Prague, Matfyzpress*, pages 298–307.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 649–652, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of* *the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers.

Zdeněk Žabokrtský. 2011. Treex-an open-source framework for natural language processing. In *ITAT*, pages 7–14.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India. Asian Federation of Natural Language Processing, International Institute of Information Technology.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. 2012. Hamledt: To parse or not to parse? In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 213–218, Marrakech, Morocco. European Language Resources Association.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. 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 1 - Volume 1*, ACL '09, pages 55–63, Stroudsburg, PA, USA. Association for Computational Linguistics.