(Re)ranking Meets Morphosyntax: State-of-the-art Results from the SPMRL 2013 Shared Task*

Anders Björkelund[§], **Özlem Çetinoğlu**[§], **Richárd Farkas**[†], **Thomas Müller**[§‡], and **Wolfgang Seeker**[§]

[§]Institute for Natural Language Processing , University of Stuttgart, Germany
[†]Department of Informatics, University of Szeged, Hungary
[‡]Center for Information and Language Processing, University of Munich, Germany
{anders,ozlem,muellets,seeker}@ims.uni-stuttgart.de
rfarkas@inf.u-szeged.hu

Abstract

This paper describes the IMS-SZEGED-CIS contribution to the SPMRL 2013 Shared Task. We participate in both the constituency and dependency tracks, and achieve state-of-the-art for all languages. For both tracks we make significant improvements through high quality preprocessing and (re)ranking on top of strong baselines. Our system came out first for both tracks.

1 Introduction

In this paper, we present our contribution to the 2013 Shared Task on Parsing Morphologically Rich Languages (MRLs). MRLs pose a number of interesting challenges to today's standard parsing algorithms, for example a free word order and, due to their rich morphology, greater lexical variation that aggravates out-of-vocabulary problems considerably (Tsarfaty et al., 2010).

Given the wide range of languages encompassed by the term MRL, there is, as of yet, no clear consensus on what approaches and features are generally important for parsing MRLs. However, developing tailored solutions for each language is time-consuming and requires a good understanding of the language in question. In our contribution to the SPMRL 2013 Shared Task (Seddah et al., 2013), we therefore chose an approach that we could apply to all languages in the Shared Task, but that would also allow us to fine-tune it for individual languages by varying certain components.

For the **dependency track**, we combined the n-best output of multiple parsers and subsequently ranked them to obtain the best parse. While this approach has been studied for constituency parsing (Zhang et al., 2009; Johnson and Ural, 2010; Wang and Zong, 2011), it is, to our knowledge, the first time this has been applied successfully within dependency parsing. We experimented with different kinds of features in the ranker and developed feature models for each language. Our system ranked first out of seven systems for all languages except French.

For the **constituency track**, we experimented with an alternative way of handling unknown words and applied a products of Context Free Grammars with Latent Annotations (PCFG-LA) (Petrov et al., 2006), whose output was reranked to select the best analysis. The additional reranking step improved results for all languages. Our system beats various baselines provided by the organizers for all languages. Unfortunately, no one else participated in this track.

For both settings, we made an effort to automatically annotate our data with the best possible preprocessing (POS, morphological information). We used a multi-layered CRF (Müller et al., 2013) to annotate each data set, stacking with the information provided by the organizers when this was beneficial. The high quality of our preprocessing considerably improved the performance of our systems.

The Shared Task involved a variety of settings as to whether gold or predicted part-of-speech tags and morphological information were available, as well as whether the full training set or a smaller (5k sen-

*Authors in alphabetical order.

Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically Rich Languages, pages 135–145,
Seattle, Washington, USA, 18 October 2013.

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
MarMoT	97.38/92.22	97.02/87.08	97.61/90.92	98.10/91.80	97.09/97.67	98.72/97.59	94.03/87.68	98.12/90.84	97.27/97.13
Stacked		98.23/89.05						98.56/92.63	97.83/97.62

Table 1: POS/morphological feature accuracies on the development sets.

tences) training set was used for training. Throughout this paper we focus on the settings with **predicted** preprocessing information with gold segmentation and the **full**[1] training sets. Unless stated otherwise, all given numbers are drawn from experiments in this setting. For all other settings, we refer the reader to the Shared Task overview paper (Seddah et al., 2013).

The remainder of the paper is structured as follows: We present our preprocessing in Section 2 and afterwards describe both our systems for the constituency (Section 3) and for the dependency tracks (Section 4). Section 5 discusses the results on the Shared Task test sets. We conclude with Section 6.

2 Preprocessing

We first spent some time on preparing the data sets, in particular we automatically annotated the data with high-quality POS and morphological information. We consider this kind of preprocessing to be an essential part of a parsing system, since the quality of the automatic preprocessing strongly affects the performance of the parsers.

Because our tools work on CoNLL09 format, we first converted the training data from the CoNLL06 format to CoNLL09. We thus had to decide whether to use coarse or fine part-of-speech (POS) tags. In a preliminary experiment we found that fine tags are the better option for all languages but Basque and Korean. For Korean the reason seems to be that the fine tag set is huge (> 900) and that the same information is also provided in the feature column.

We predict POS tags and morphological features jointly using the Conditional Random Field (CRF) tagger MarMoT[2] (Müller et al., 2013).

MarMoT incrementally creates forward-backward lattices of increasing order to prune the sizable space of possible morphological analyses. We use MarMoT with the default parameters.

Since morphological dictionaries can improve automatic POS tagging considerably, we also created such dictionaries for each language. For this, we analyzed the word forms provided in the data sets with language-specific morphological analyzers except for Hebrew and German where we just extracted the morphological information from the lattice files provided by the organizers. For the other languages we used the following tools: Arabic: AraMorph a reimplementation of Buckwalter (2002), Basque: Apertium (Forcada et al., 2011), French: an IMS internal tool,[3] Hungarian: Magyarlanc (Zsibrita et al., 2013), Korean: HanNanum (Park et al., 2010), Polish: Morfeusz (Woliński, 2006), and Swedish: Granska (Domeij et al., 2000).

The created dictionaries were shared with the other Shared Task participants. We used these dictionaries as additional features for MarMoT.

For some languages we also integrated the predicted tags provided by the organizers into the feature model. These *stacked* models gave improvements for Swedish, Polish and Basque (cf. Table 1 for accuracies).

For the full setting the training data was annotated using 5-fold jackknifing. In the 5k setting, we additionally added all sentences not present in the parser training data to the training data sets of the tagger. This is similar to the predicted 5k files provided by the organizers, where more training data than the 5k was also used for prediction.

Table 3 presents a comparison between our graph-based baseline parser using the preprocessing explained in this section (denoted mate) and the preprocessing provided by the organizers (denoted mate'). Our preprocessing yields improvements for all languages but Swedish. The worse performance for Swedish is due to the fact that the predictions provided by the organizers were produced by models that were trained on a much larger data

[1] Although, for Hebrew and Swedish only 5k sentences were available for training, and the two settings thus coincide.

[2] `https://code.google.com/p/cistern/`

[3] The French morphology was written by Zhenxia Zhou, Max Kisselew and Helmut Schmid. It is an extension of Zhou (2007) and implemented in SFST (Schmid, 2005).

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Berkeley	78.24	69.17	79.74	81.74	87.83	83.90	70.97	84.11	74.50
Replaced	78.70	84.33	79.68	82.74	89.55	89.08	82.84	87.12	75.52
Product	80.30	86.21	81.42	84.56	**90.49**	89.80	84.15	88.32	79.25
Reranked	**81.24**	**87.35**	**82.49**	**85.01**	**90.49**	**91.07**	**84.63**	**88.40**	**79.53**

Table 2: PARSEVAL scores on the development sets.

set. The comparison with other parsers demonstrates that for some languages (e.g., Hebrew or Korean) the improvements due to better preprocessing can be greater than the improvements due to a better parser. For instance, for Hebrew the parser trained on the provided preprocessing is more than three points (LAS) behind the three parsers trained on our own preprocessing. However, the difference between these three parsers is less than a point.

3 Constituency Parsing

The phrase structure parsing pipeline is based on products of Context Free Grammars with Latent Annotations (PCFG-LA) (Petrov et al., 2006) and discriminative reranking. We further replace rare words by their predicted morphological analysis.

We preprocess the treebank trees by removing the morphological annotation of the POS tags and the function labels of all non-terminals. We also reduce the 177 compositional Korean POS tags to their first atomic tag, which results in a POS tag set of 9 tags.

PCFG-LAs are incrementally built by splitting non-terminals, refining parameters using EM-training and reversing splits that only cause small increases in likelihood.

Running the Berkeley Parser[4] – the reference implementation of PCFG-LAs – on the data sets results in the PARSEVAL scores given in Table 2 (Berkeley). The Berkeley parser only implements a simple signature-based unknown word model that seems to be ineffective for some of the languages, especially Basque and Korean.

We thus replace rare words (frequency < 20) by the predicted morphological tags of Section 2 (or the true morphological tag for the gold setup). The intuition is that our discriminative tagger has a more sophisticated unknown word treatment than the Berkeley parser, taking for example prefixes, suffixes and the immediate lexical context into account. Furthermore, the morphological tag contains most of the necessary syntactic information. An exception, for instance, might be the semantic information needed to disambiguate prepositional attachment. We think that replacing rare words by tags has an advantage over constraining the pre-terminal layer of the parser, because the parser can still decide to assign a different tag, for example in cases were the tagger produces errors due to long-distance dependencies. The used frequency threshold of 20 results in token replacement rates of 18% (French) to 57% (Korean and Polish), which correspond to 209 (for Polish) to 3221 (for Arabic) word types that are not replaced. The PARSEVAL scores for the described method are again given in Table 2 (Replaced). The method yields improvements for all languages except for French where we observe a drop of 0.06. The improvements range from 0.46 for Arabic to 1.02 for Swedish, 3.1 for Polish and more than 10 for Basque and Korean.

To further improve results, we employ the product-of-grammars procedure (Petrov, 2010), where different grammars are trained on the same data set but with different initialization setups. We trained 8 grammars and used tree-level inference. In Table 2 (Product) we can see that this leads to improvements from 0.72 for Hungarian to 3.73 for Swedish.

On the 50-best output of the product parser, we also carry out discriminative reranking. The reranker is trained for the maximum entropy objective function of Charniak and Johnson (2005) and use the standard feature set – without language-specific feature engineering – from Charniak and Johnson (2005) and Collins (2000). We use a slightly modified version of the Mallet toolkit (McCallum, 2002) for reranking.

Improvements range from negligible differences ($< .1$) for Hebrew and Polish to substantial differences ($> 1.$) for Basque, French, and Hungarian.

[4] http://code.google.com/p/berkeleyparser/

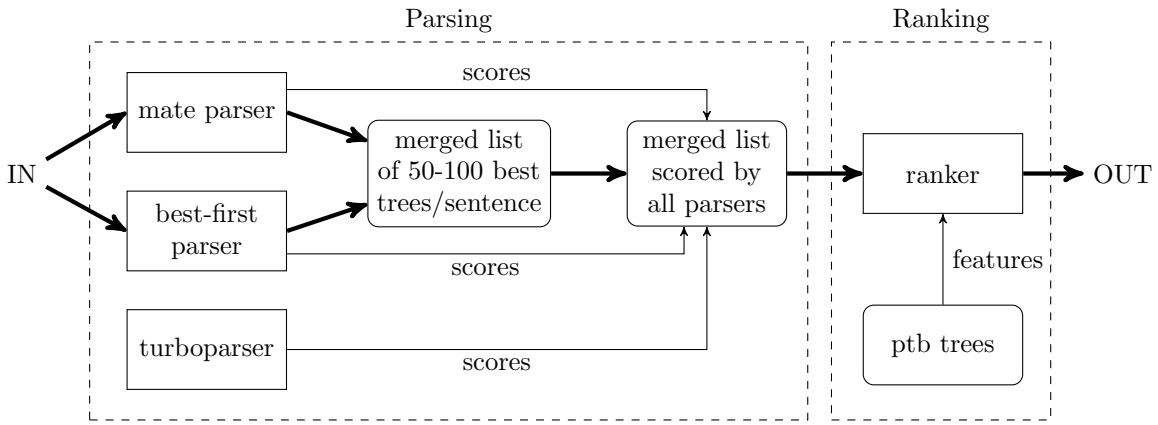

Figure 1: Architecture of the dependency ranking system.

For our final submission, we used the reranker output for all languages except French, Hebrew, Polish, and Swedish. This decision was based on an earlier version of the evaluation setting provided by the organizers. In this setup, reranking did not help or was even harmful for these four languages. The figures in Table 2 use the latest evaluation script and are thus consistent with the test set results presented in Section 5.

After the submission deadline the Shared Task organizers made us aware that we had surprisingly low exact match scores for Polish (e.g., 1.22 for the reranked setup). The reason seems to be that the Berkeley parser cannot produce unary chains of length > 2. The gold development set contains 1783 such chains while the prediction of the reranked system contains none. A particularly frequent unary chain with 908 occurences in the gold data is ff $\rightarrow$ fwe $\rightarrow$ formaczas. As this chain cannot be produced the parser leaves out the fwe phrase. Inserting new fwe nodes between ff and formacszas nodes raises the PARSEVAL scores of the reranked model from 88.40 to 90.64 and the exact match scores to 11.34. This suggests that the Polish results could be improved substantially if unary chains were properly dealt with, for example by collapsing unary chains.[5]

4 Dependency Parsing

The core idea of our dependency parsing system is the combination of the n-best output of several parsers followed by a ranking step on the combined list. Specifically, we first run two parsers that each output their 50-best analyses for each sentence. These 50-best analyses are merged together into one single n-best list of between 50 and 100 analyses (depending on the overlap between the n-best lists of the two parsers). We then use the two parsers plus an additional one to score each tree in the n-best lists according to their parsing model, thus providing us with three different scores for each tree in the n-best lists. The n-best lists are then given to a ranker, which ranks the list using the three scores and a small set of additional features in order to find the best overall analysis. Figure 1 shows a schematic of the process.

As a preprocessing step, we reduced the dependency label set for the Hungarian training data. The Hungarian dependency data set encodes ellipses through composite edge labels which leads to a proliferation of edge labels (more than 400). Since many of these labels are extremely rare and thus hard to learn for the parsers, we reduced the set of edge labels during the conversion. Specifically, we retained the 50 most frequent labels, while reducing the composite labels to their base label.

For producing the initial n-best lists, we use the mate parser[6] (Bohnet, 2010) and a variant of the EasyFirst parser (Goldberg and Elhadad, 2010), which we here call best-first parser.

The mate parser is a state-of-the-art graph-based dependency parser that uses second-order features.

[5] Thanks to Slav Petrov for pointing us to the unary chain length limit.

[6] https://code.google.com/p/mate-tools

The parser works in two steps. First, it uses dynamic programming to find the optimal projective tree using the Carreras (2007) decoder. It then applies the non-projective approximation algorithm proposed by McDonald and Pereira (2006) in order to produce non-projective parse trees. The non-projective approximation algorithm is a greedy hill climbing algorithm that starts from the optimal projective parse and iteratively tries to reattach all tokens, one at a time, everywhere in the sentence as long as the tree property holds. It halts when the increase in the score of the tree according to the parsing model is below a certain threshold.

n-best lists are obtained by applying the non-projective approximation algorithm in a non-greedy manner, exploring multiple possibilities. All trees are collected in a list, and when no new trees are found, or newer trees have a significantly lower score than the currently best one, search halts. The n best trees are then retrieved from the list. It should be noted that, in the standard case, the non-projective approximation algorithm may find a local optimum, and that there may be other trees that have a higher score which were not explored. Thus the best parse in the greedy case may not necessarily be the one with the highest score in the n-best list. Since the parser is trained with the greedy version of the non-projective approximation algorithm, the greedily chosen output parse tree is of special interest. We thus flag this tree as the **baseline mate parse**, in order to use that for features in the ranker. The baseline mate parse is also our overall baseline in the dependency track.

The best-first parser deviates from the EasyFirst parser in several small respects: The EasyFirst decoder creates dependency links between the roots of adjacent substructures, which gives an $O(n \log n)$ complexity, but restricts the output to projective trees. The best-first parser is allowed to choose as head any node of an adjacent substructure instead of only the root, which increases complexity to $O(n^2)$, but accounts for a big part of possible non-projective structures. We additionally implemented a swap-operation (Nivre, 2009; Tratz and Hovy, 2011) to account for the more complex structures. The best-first parser relies on a beam-search strategy[7] to pursue multiple derivations, which we also use to produce the n-best output.

In the scoring step, we additionally apply the turboparser[8] (Martins et al., 2010), which is based on linear programming relaxations.[9] We changed all three parsers such that they would return a score for a given tree. We use this to extract scores from each parser for all trees in the n-best lists. It is important to have a score from every parser for every tree, as previously observed by Zhang et al. (2009) in the context of constituency reranking.

4.1 Ranking

Table 3 shows the performance of the individual parsers measured on the development sets. It also displays the **oracle** scores over the different n-best lists, i.e., the maximal possible score over an n-best list if the best tree is always selected.

The mate parser generally performs best followed by turboparser, while the best-first parser comes last. But we can see from the oracle scores that the best-first parser often shows comparable or even higher oracle scores than mate, and that the combination of the n-best lists always adds substantial improvements to the oracle scores. These findings show that the mate and best-first parsers are providing different sets of n-best lists. Moreover, all three parsers rely on different parsing algorithms and feature sets. For these reasons, we hypothesized that the parsers contribute different views on the parse trees and that their combination would result in better overall performance.

In order to leverage the diversity between the parsers we experimented with **ranking**[10] on the n-best lists. We used the same ranking model introduced in Section 3 here as well. The model is trained to select the best parse according to the labeled attachment score (LAS). The training data for the ranker was created by 5-fold jackknifing on the training sets. The feature sets for the ranker for

[7] Due to the nature of the decoder, the parser can produce spurious ambiguities in the beam. If this occurs, only the one with the higher score is kept.

[8] `http://www.ark.cs.cmu.edu/TurboParser/`

[9] Ideally we would also extract n-best lists from the turboparser, however time prevented us from making the necessary modifications.

[10] We refrain from calling it *re*ranking in this setting, since we are using merged n-best lists and the initial ranking is not entirely clear to begin with.

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Baseline results for individual parsers									
mate'		88.50/83.50	88.18/84.49	92.71/90.85	83.63/75.89	87.07/82.84	86.06/82.39	91.17/85.81	83.65/77.16
mate	87.68/85.42	89.11/84.43	88.30/84.84	93.15/91.46	86.05/79.37	88.03/84.41	87.91/85.76	91.51/86.30	83.53/77.05
bf	87.61/85.32	84.07/75.90	87.45/83.92	92.90/91.10	86.10/79.57	83.85/75.94	86.54/83.97	90.10/83.75	82.27/75.36
turbo	87.82/85.35	88.88/83.84	88.24/84.57	93.59/91.54	85.74/78.95	86.86/82.80	88.35/86.23	90.97/85.55	83.24/76.15
Oracle scores for n-best lists									
mate	90.85/88.74	93.39/89.85	90.99/87.81	97.14/95.84	89.05/83.03	91.41/88.19	94.86/92.96	95.19/91.67	87.19/81.66
bf	91.47/89.46	91.68/86.46	91.38/88.68	97.40/96.60	91.04/85.67	87.64/81.79	94.90/92.94	96.25/93.74	87.60/82.46
merged	92.65/90.71	95.15/91.91	92.97/90.43	98.19/97.44	92.39/87.18	92.12/88.76	96.23/94.65	97.28/95.29	89.87/84.96

Table 3: Baseline performance and n-best oracle scores (UAS/LAS) on the development sets. mate' uses the preprocessing provided by the organizers, the other parsers use the preprocessing described in Section 2.

each language were optimized manually via cross-validation on the training sets. The features used for each language, as well as a default (baseline) feature set, are shown in Table 4. We now outline the features we used in the ranker:

Score from the base parsers – denoted **B**, **M**, **T**, for the best-first, mate, and turbo parsers, respectively. We also have indicator features whether a certain parse was the best according to a given parser, denoted **GB**, **GM**, **GT**, respectively. Since the mate parser does not necessarily assign the highest score to the baseline mate parse, the GM feature is a ternary feature which indicates whether a parse is the same as the baseline mate parse, or better, or worse. We also experimented with transformations and combinations of the scores from the parsers. Specifically, **BMProd** denotes the product of B and M; **BMeProd** denotes the sum of B and M in e-space, i.e., e^{B+M}; **reBMT**, **reBT**, **reMT** denote the normalized product of the corresponding scores, where scores are normalized in a softmax fashion such that all features take on values in the interval $(0, 1)$.

Projectivity features (Hall et al., 2007) – the number of non-projective edges in a tree, denoted **np**. Whether a tree is ill-nested, denoted **I**. Since ill-nested trees are extremely rare in the treebanks, this helps the ranker filter out unlikely candidates from the n-best lists. For a definition and further discussion of ill-nestedness, we refer to (Havelka, 2007).

Constituent features – from the constituent track we also have constituent trees of all sentences which can be used for feature extraction. Specifically, for every head-dependent pair, we extract the path in the constituent tree between the nodes, denoted **ptbp**.

Case agreement – on head-dependent pairs that both have a case value assigned among their morphological features, we mark whether it is the same case or not, denoted **case**.

Function label uniqueness – on each training set we extracted a list of function labels that generally occur at most once as the dependent of a node, e.g., subjects or objects. Features are then extracted from all nodes that have one or more dependents of each label aimed at capturing mistakes such as double subjects on a verb. This template is denoted **FL**.

In addition to the features mentioned above, we experimented with a variety of feature templates, including features drawn from previous work on dependency reranking (Hall, 2007), e.g., lexical and POS-based features over edges, "subcategorization" frames (i.e., the concatenation of POS-tags that are headed by a certain node in the tree), etc, although these features did not seem to help. For German we created feature templates based on the constraints used in the constraint-based parser by Seeker and Kuhn (2013). This includes, e.g., violations in case or number agreement between heads and dependents, as well as more complex features that consider labels on entire verb complexes. None of these features yielded any clear improvements though. We also experimented with features that target some specific constructions (and specifics of annotation schemes) which the parsers typically cannot fully see, such as coordination, however, also here we saw no clear improvements.

4.2 Effects of Ranking

In Table 5, we show the improvements from using the ranker, both with the baseline and optimized features sets for the ranker. For the sake of comparison,

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Baseline	87.68/85.42	89.11/84.43	88.30/84.84	93.15/91.46	86.05/79.37	88.03/84.41	87.91/85.76	91.51/86.30	83.53/77.05
Ranked-dflt	88.54/86.32	**89.99**/85.43	88.85/85.39	94.06/92.36	87.28/80.44	88.16/84.54	88.71/86.65	92.26/87.12	84.51/77.83
Ranked	**88.93/86.74**	89.95/**85.61**	**89.37/85.96**	**94.20/92.68**	**87.63/81.02**	**88.38/84.77**	**89.20/87.12**	**93.02/87.69**	**85.04/78.57**
Oracle	92.65/90.71	95.15/91.91	92.97/90.43	98.19/97.44	92.39/87.18	92.12/88.76	96.23/94.65	97.28/95.29	89.87/84.96

Table 5: Performance (UAS/LAS) of the reranker on the development sets. Baseline denotes our baseline. Ranked-dflt and Ranked denote the default and optimized ranker feature sets, respectively. Oracle denotes the oracle scores.

default	B, M, T, GB, GM, GT, I
Arabic	B, M, T, GB, GM, I, ptbp, reBMT
Basque	B, M, T, GB, GM, GT, I, ptbp, I, reMT, case
French	B, M, T, GB, GM, GT, I, ptbp
German	B, M, T, GM, I, BMProd, FL
Hebrew	B, M, T, GB, GM, GT, I, ptbp, FL, BMeProd
Hungarian	B, M, T, GB, GM, GT, I, ptbp, reBM, FL
Korean	B, M, T, GB, GM, GT, I, ptbp, reMT, FL
Polish	B, M, T, GB, GM, GT, I, ptbp, np
Swedish	B, M, T, GB, GM, GT, I, ptbp, reBM, FL

Table 4: Feature sets for the dependency ranker for each language. default denotes the default ranker feature set.

the baseline mate parses as well as the oracle parses on the merged n-best lists are repeated from Table 3. We see that ranking clearly helps, both with a tailored feature set, as well as the default feature set. The improvement in LAS between the baseline and the tailored ranking feature sets ranges from 1.1% (French) to 1.6% (Hebrew) absolute, with the exception of Hungarian, where improvements on the dev set are more modest (contrary to the test set results, cf. Section 5). Even with the default feature set, the improvements range from 0.5% (French) to 1.1% (Hebrew) absolute, again setting Hungarian aside. We believe that this is an interesting result considering the simplicity of the default feature set.

5 Test Set Results

In this section we outline our final results on the test sets. As previously, we focus on the setting with predicted tags in gold segmentation and the largest training set. We also present results on Arabic and Hebrew for the predicted segmentation setting. For the gold preprocessing and all 5k settings, we refer the reader to the Shared Task overview paper (Seddah et al., 2013).[11]

In Table 7, we present our results in the constituency track. Since we were the only participating team in the constituency track, we compare ourselves with the best baseline[12] provided by the organizers. Our system outperforms the baseline for all languages in terms of PARSEVAL F_1. Following the trend on the development sets, reranking is consistently helping across languages.[13] Despite the lack of other submissions in the shared task, we believe our numbers are generally strong and hope that they can serve as a reference for future work on constituency parsing on these data sets.

Table 8 displays our results in the dependency track. We submitted two runs: a baseline, which is the baseline mate parse, and the reranked trees. The table also compares our results to the best performing other participant in the shared task (denoted Other) as well as the MaltParser (Nivre et al., 2007) baseline provided by the shared task organizers (denoted ST Baseline). We obtain the highest scores for all languages, with the exception of French. It is also clear that we make considerable gains over our baseline, confirming our results on the development sets reported in Section 4. It is also noteworthy that our baseline (i.e., the mate parser with our own preprocessing) outperforms the best other system for 5 languages.

	Arabic	Hebrew
Other	90.75/8.48	88.33/12.20
Dep. Baseline	91.13/9.10	89.27/15.01
Dep. Ranked	91.74/9.83	89.47/16.97
Constituency	92.06/9.49	89.30/13.60

Table 6: Unlabeled TedEval scores (accuracy/exact match) for the test sets in the predicted segmentation setting. Only sentences of length ≤ 70 are evaluated.

[11] Or the results page online: `http://www.spmrl.org/spmrl2013-sharedtask-results.html`

[12] It should be noted that the Shared Task organizers computed 2 different baselines on the test sets. The best baseline results for each language thus come from different parsers.

[13] We remind the reader that our submission decisions are not based on figures in Table 2, cf. Section 3.

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
ST Baseline	79.19	74.74	80.38	78.30	86.96	85.22	78.56	86.75	80.64
Product	80.81	87.18	81.83	80.70	89.46	90.58	83.49	87.55	83.99
Reranked	**81.32**	**87.86**	**82.86**	**81.27**	**89.49**	**91.85**	**84.27**	**87.76**	**84.88**

Table 7: Final PARSEVAL F_1 scores for constituents on the test set for the predicted setting. ST Baseline denotes the best baseline (out of 2) provided by the Shared Task organizers. Our submission is underlined.

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
ST Baseline	83.18/80.36	79.77/70.11	82.49/77.98	81.51/77.81	76.49/69.97	80.72/70.15	85.72/82.06	82.19/75.63	80.29/73.21
Other	85.78/83.20	89.19/84.25	**89.19/85.86**	90.80/88.66	81.05/73.63	88.93/84.97	85.84/82.65	88.12/82.56	87.28/80.88
Baseline	86.96/84.81	89.32/84.25	87.87/84.37	90.54/88.37	85.88/79.67	89.09/85.31	87.41/85.51	90.30/85.51	86.85/80.67
Ranked	**88.32/86.21**	**89.88/85.14**	88.68/85.24	**91.64/89.65**	**86.70/80.89**	**89.81/86.13**	**88.47/86.62**	**91.75/87.07**	**88.06/82.13**

Table 8: Final UAS/LAS scores for dependencies on the test sets for the predicted setting. Other denotes the highest scoring other participant in the Shared Task. ST Baseline denotes the MaltParser baseline provided by the Shared Task organizers.

Table 6 shows the unlabeled TedEval (Tsarfaty et al., 2012) scores (accuracy/exact match) on the test sets for the predicted segmentation setting for Arabic and Hebrew. Note that these figures only include sentences of length less than or equal to 70. Since TedEval enables cross-framework comparison, we compare our submissions from the dependency track to our submission from the constituency track. In these runs we used the same systems that were used for the gold segmentation with predicted tags track. The predicted segmentation was provided by the Shared Task organizers. We also compare our results to the best other system from the Shared Task (denoted Other).

Also here we obtain the highest results for both languages. However, it is unclear what syntactic paradigm (dependencies or constituents) is better suited for the task. All in all it is difficult to assess whether the differences between the best and second best systems for each language are meaningful.

6 Conclusion

We have presented our contribution to the 2013 SPMRL Shared Task. We participated in both the constituency and dependency tracks. In both tracks we make use of a state-of-the-art tagger for POS and morphological features. In the constituency track, we use the tagger to handle unknown words and employ a product-of-grammars-based PCFG-LA parser and parse tree reranking. In the dependency track, we combine multiple parsers output as input for a ranker.

Since there were no other participants in the constituency track, it is difficult to draw any conclusions from our results. We do however show that the application of product grammars, our handling of rare words, and a subsequent reranking step outperforms a baseline PCFG-LA parser.

In the dependency track we obtain the best results for all languages except French among 7 participants. Our reranking approach clearly outperforms a baseline graph-based parser. This is the first time multiple parsers have been used in a dependency reranking setup.

Aside from minor decisions made on the basis of each language, our approach is language agnostic and does not target morphology in any particular way as part of the parsing process. We show that with a strong baseline and with no language specific treatment it is possible to achieve state-of-the-art results across all languages. Our architecture for the dependency parsing track enables the use of language-specific features in the ranker, although we only had minor success with features that target morphology. However, it may be the case that approaches from previous work on parsing MRLs, or the approaches taken by other teams in the Shared Task, can be successfully combined with ours and improve parsing accuracy even more.

Acknowledgments

Richárd Farkas is funded by the European Union and the European Social Fund through the project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-

2012-0013). Thomas Müller is supported by a Google Europe Fellowship in Natural Language Processing. The remaining authors are funded by the Deutsche Forschungsgemeinschaft (DFG) via the SFB 732, projects D2 and D8 (PI: Jonas Kuhn).

We also express our gratitude to the treebank providers for each language: Arabic (Maamouri et al., 2004; Habash and Roth, 2009; Habash et al., 2009; Green and Manning, 2010), Basque (Aduriz et al., 2003), French (Abeillé et al., 2003), Hebrew (Sima'an et al., 2001; Tsarfaty, 2010; Goldberg, 2011; Tsarfaty, 2013), German (Brants et al., 2002; Seeker and Kuhn, 2012), Hungarian (Csendes et al., 2005; Vincze et al., 2010), Korean (Choi et al., 1994; Choi, 2013), Polish (Świdziński and Woliński, 2010), and Swedish (Nivre et al., 2006).

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. In *TLT-03*, pages 201–204.

Bernd Bohnet. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In Erhard Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 24–41, Sozopol, Bulgaria.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. *Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.*

Xavier Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, Prague, Czech Republic, June. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180.

Key-Sun Choi, Young S Han, Young G Han, and Oh W Kwon. 1994. Kaist tree bank project for korean: Present and future development. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 7–14. Citeseer.

Jinho D. Choi. 2013. Preparing korean data for the shared task on parsing morphologically rich languages. *ArXiv e-prints*.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Dóra Csendes, Janós Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged treebank. In Václav Matoušek, Pavel Mautner, and Tomáš Pavelka, editors, *Text, Speech and Dialogue: Proceedings of TSD 2005*. Springer.

Rickard Domeij, Ola Knutsson, Johan Carlberger, and Viggo Kann. 2000. Granska-an efficient hybrid system for Swedish grammar checking. In *In Proceedings of the 12th Nordic Conference in Computational Linguistics*.

Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. 2011. Apertium: A free/open-source platform for rule-based machine translation. *Machine Translation*.

Yoav Goldberg and Michael Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, California, June. Association for Computational Linguistics.

Yoav Goldberg. 2011. *Automatic syntactic processing of Modern Hebrew*. Ph.D. thesis, Ben Gurion University of the Negev.

Spence Green and Christopher D. Manning. 2010. Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, Beijing, China, August. Coling 2010 Organizing Committee.

Nizar Habash and Ryan Roth. 2009. Catib: The columbia arabic treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore, August. Association for Computational Linguistics.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Keith Hall, Jiri Havelka, and David A. Smith. 2007. Log-Linear Models of Non-Projective Trees, k-best MST Parsing and Tree-Ranking. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 962–966, Prague, Czech Republic, June. Association for Computational Linguistics.

Keith Hall. 2007. K-best Spanning Tree Parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, Prague, Czech Republic, June. Association for Computational Linguistics.

Jiri Havelka. 2007. Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615, Prague, Czech Republic, June. Association for Computational Linguistics.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the Berkeley and Brown Parsers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 665–668, Los Angeles, California, June. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*.

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA, October. Association for Computational Linguistics.

Andrew Kachites McCallum. 2002. "mallet: A machine learning for language toolkit". http://mallet.cs.umass.edu.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy. Association for Computational Linguistics.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient Higher-Order CRFs for Morphological Tagging. In *In Proceedings of EMNLP*.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proceedings of LREC*, pages 1392–1395, Genoa, Italy.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 6.

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. 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*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

S Park, D Choi, E-k Kim, and KS Choi. 2010. A plug-in component-based Korean morphological analyzer. In *Proceedings of HCLT2010*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Slav Petrov. 2010. Products of Random Latent Variable Grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Los Angeles, California, June. Association for Computational Linguistics.

Helmut Schmid. 2005. A programming language for finite state transducers. In *FSMNLP*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, and Alina Wróblewska. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, Seattle, WA.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and Syntactic Case in Statistical Dependency Parsing. *Computational Linguistics*, 39(1):23–55.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In *Text, Speech and Dialogue: 13th International Conference (TSD)*, Lecture Notes in Artificial Intelligence, pages 197—204, Brno, Czech Republic. Springer.

Stephen Tratz and Eduard Hovy. 2011. A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1268, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical Parsing of Morphologically Rich Languages (SPMRL) What, How and Whither. In *Proc. of the SPMRL Workshop of NAACL-HLT*, pages 1–12, Los Angeles, CA, USA.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012. Joint Evaluation of Morphological Segmentation and Syntactic Parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 6–10, Jeju Island, Korea, July. Association for Computational Linguistics.

Reut Tsarfaty. 2010. *Relational-Realizational Parsing*. Ph.D. thesis, University of Amsterdam.

Reut Tsarfaty. 2013. *A Unified Morpho-Syntactic Scheme of Stanford Dependencies*. Proceedings of ACL.

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *LREC*.

Zhiguo Wang and Chengqing Zong. 2011. Parse Reranking Based on Higher-Order Lexical Dependencies. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1251–1259, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Marcin Woliński. 2006. Morfeusz - A practical tool for the morphological analysis of Polish. In *Intelligent information processing and web mining*, pages 511–520. Springer.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-Best Combination of Syntactic Parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1552–1560, Singapore, August. Association for Computational Linguistics.

Zhenxia Zhou. 2007. Entwicklung einer französischen Finite-State-Morphologie. Diplomarbeit, Institute for Natural Language Processing, University of Stuttgart.

János Zsibrita, Veronika Vincze, and Richárd Farkas. 2013. Magyarlanc 2.0: Szintaktikai elemzés és felgyorsított szófaji egyértelműsítés. In *IX. Magyar Számítógépes Nyelvészeti Konferencia*.