The IMS-Wrocław-Szeged-CIS Entry at the SPMRL 2014 Shared Task: Reranking and Morphosyntax Meet Unlabeled Data*

Anders Björkelund[§] and **Özlem Çetinoğlu**[§] and **Agnieszka Faleńska**[◇,§]

Richárd Farkas[†] and **Thomas Müller**[‡] and **Wolfgang Seeker**[§] and **Zsolt Szántó**[†]

[§]Institute for Natural Language Processing, University of Stuttgart, Germany
[◇]Institute of Computer Science, University of Wrocław, Poland
[†]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
agnieszka.falenska@cs.uni.wroc.pl
{rfarkas,szantozs}@inf.u-szeged.hu

Abstract

This paper describes our contribution to the SPMRL 2014 Shared Task. We participated in the predicted POS and morphology setting using full-size training data, and for all languages except Arabic. Our approach builds upon our contribution from last year (Björkelund et al., 2013), with additions that utilize unlabeled data. We observed that exploiting unlabeled data is challenging and we could benefit from it only moderately. We achieved best scores on all languages in the dependency track and on all languages except Polish in the constituency track.

1 Introduction

We present our contribution to the SPMRL 2014 Shared Task on parsing morphologically rich languages (Seddah et al., 2014). This year's shared task (ST) is a direct extension of the SPMRL 2013 Shared Task (Seddah et al., 2013) which involved parsing both constituency and dependency representations of 9 languages: Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, and Swedish. The 2014 ST extends the task by making large amounts of unlabeled data available to participants.

Our contribution builds upon our system from last year (Björkelund et al., 2013), and extends it with additional features and components. We started with our tried and tested methods and made some design decisions (e.g., use of coarse or fine grained POS tags) based on previous experience to reduce our experimental space. We then explored new techniques, some of which make use of unlabeled data.

Following last year, we predict our own POS tags and morphological features and integrate output of external morphological analyzers and organizer-provided predictions as features. We change our integration policy to make use of unlabeled data, which results in lower preprocessing accuracies than previous year. We compensate this drop in the parsing step and achieve competitive or better results.

In our 2013 submission on the **dependency track**, we combined n-best output of multiple parsers and ranked them. This year, we extend our system by using supertags (Bangalore and Joshi, 1999) in parsing (Ouchi et al., 2014; Ambati et al., 2014), adding a blended tree (Sagae and Lavie, 2006) to the n-best list, and utilizing unlabeled data as features in reranking.

Our contribution to the 2013 **constituency track** handled lexical sparsity by removing morphological annotation of POS tags and replacing rare words with their morphological analysis. This year, we also experiment with an extended lexicon model (Goldberg and Elhadad, 2013). We then apply a product grammar (Petrov, 2010) and a reranker (Charniak and Johnson, 2005), and enrich our ranking features with Brown clusters (Brown et al., 1992) and atomic morphological features (Szántó and Farkas, 2014).

*Authors in alphabetical order

Given the limited window of time to participate in this year's shared task, we only contribute to the setting with *predicted* preprocessing, using the largest available training data set for each language.[1] We also do not participate in the Arabic track due to late availability of unlabeled data. For the remaining languages, we achieved the best scores in both constituency and dependency tracks with the exception of Polish constituency parsing.

2 Preprocessing

As the initial step of preprocessing we converted the Shared Task data from the CoNLL06 format to CoNLL09, which requires a decision on whether to use coarse or fine grained POS tags. Based on our 2013 ST experiments we prefer fine POS tags where possible, except for Basque and Korean.

We use MarMoT[2] (Müller et al., 2013) to predict POS tags and morphological features jointly. We extracted word forms from the training, development, and unlabeled data and analyzed them with language-specific morphological analyzers for six languages.[3] We integrate the output from these morphological analyzers as features in MarMoT. We shared these morphological dictionaries with other participants through the ST organizers.

We also utilize the predicted tags provided by the organizers by creating dictionaries which list all the predictions of a word form that are present in the training, development and unlabeled data files. This is a more indirect way of using organizer-provided predictions than stacking (which we used last year), as in the latter only its best prediction is assigned to a word form. Despite its indirectness we opt for the dictionary approach to deal with the label inconsistencies on the labeled and unlabeled data for some languages.[4]

We use the mate-tools' lemmatizer[5] both on the treebanks and unlabeled data. For both POS and morphological feature tagging and lemmatization, we did 5-fold jackknifing on the training data to produce realistic input features for the parsers.

Table 1 gives the POS and morphological feature accuracies on the development sets. We observe POS accuracy drops ranging from 0.12 (Hebrew, German) to 0.71 (Basque) and drops in morphological accuracies ranging from 0.14 (Hungarian) to 1.63 (Polish) as compared to last year.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Björkelund et al. 2013	*98.23/89.05*	97.61/90.92	98.10/91.80	97.09/97.67	98.72/97.59	94.03/87.68	*98.56/92.63*	*97.83/97.62*
This year	97.52/87.81	97.08/89.36	97.98/90.38	96.97/97.15	98.49/97.45	93.82/87.44	98.39/91.00	97.40/97.16

Table 1: MarMoT POS/morphological feature accuracies on the development sets from last year and this year. Italicized figures denote languages where stacking was used last year (Basque, Polish, Swedish).

3 Constituency Parsing

Our constituency parsing architecture consists of two steps. First, we deal with lexical sparsity and exploit product grammars. Second, we apply a reranker where we investigate new feature templates. In the following sections we focus on the methods to alleviate lexical sparsity and features we use in the reranker.

3.1 Lexical Sparsity

The out-of-vocabulary issue is a crucial problem in morphologically rich languages, as a word can have many different forms depending on its syntactic and semantic context. Last year, we replaced rare words

[1] In other words, no gold preprocessing or smaller training sets.

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

[3] Basque: Apertium (Forcada et al., 2011), French: an IMS internal tool by Zhenxia Zhou, Max Kisselew and Helmut Schmid. It is an extension of Zhou (2007) and implemented in SFST (Schmid, 2005). German: SMOR (Schmid et al., 2004) Hungarian: Magyarlanc (Zsibrita et al., 2013), Korean: HanNanum (Park et al., 2010), Polish: Morfeusz (Woliński, 2006).

[4] The organizers later resolved the issue by patching the data, but time constraints prevented us from using the patched data.

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

by their morphological analysis produced by MarMoT (similar to the strategy of backing off rare words to their POS tag in the CCG literature (Clark and Curran, 2007)). We call this strategy *Replace*.

This year, we experiment with an alternative approach, which exploits the available unlabeled data. We followed Goldberg and Elhadad (2013) and enhanced a lexicon model trained on the treebank training data with frequency information about the possible morphological analyses of tokens (*ExtendLex*). We estimate the tagging probability $P(t|w)$ of the tag t given the word w by

$$P(t|w) = \begin{cases} P_{tb}(t|w), & \text{if } c(w) \geq K \\ \frac{c(w)P_{tb}(t|w)+P_{ex}(t|w)}{1+c(w)}, & \text{otherwise} \end{cases}$$

where $c(w)$ is the count of w in the training set, K is a predefined constant, $P_{tb}(t|w)$ is the probability estimate from the treebank (the relative frequency with smoothing) and $P_{ex}(t|w)$ is the probability estimate from an external lexicon. We calculate the emission probabilities $P(w|t)$ from the tagging probabilities $P(t|w)$ by applying Bayes' rule. We construct the external lexicon by relative frequencies of MarMoT's morphological tagging counted on the development and unlabeled data, and we use the default value $K = 7$ for all languages (for details see Szántó and Farkas (2014)).

We note that the two strategies lead to fundamentally different representations. In the *Replace* version the output parses contain morphological descriptions instead of tokens and only main POS tags are used as preterminal labels while in the *ExtendLex* approach tokens remain at the terminal level and full morphological analyses are employed as preterminal labels.[6]

Table 2 shows the results achieved by the two strategies on the development sets. As our baselines we use the **Berkeley parser** (Petrov et al., 2006) by removing morphological annotations and leaving only POS tags in preterminals (**mainPOS**), and by using full morphological descriptions (**fullMorph**). For German and Swedish, where the main POS is significantly better than the full morphological description, *ExtendLex* gets lower scores than *Berkeley*$_{mainPOS}$, but the morphological information can help later in the reranking step. For all languages *Replace* outperforms *ExtendLex*, but when we adopt the products of respective grammars (Petrov, 2010), *ExtendLex Product* achieves slightly better scores than *Replace Product* for French and Hebrew.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Berkeley$_{mainPOS}$	72.32	79.35	82.26	88.71	83.84	71.85	86.75	75.19
Berkeley$_{fullMorph}$	77.82	79.17	80.22	88.40	87.18	82.28	85.06	72.82
ExtendLex	77.51	79.67	81.54	89.33	88.99	-	88.21	74.57
Replace	84.27	80.26	82.99	89.73	89.59	83.07	90.29	77.08
ExtendLex Product	80.71	**81.38**	82.13	**89.92**	90.43	-	91.52	78.21
Replace Product	**85.31**	81.29	**84.55**	89.87	**90.72**	**83.86**	**92.28**	**78.66**

Table 2: PARSEVAL scores on the development sets for the predicted setting. Best results for each language are displayed in bold.

We tried several strategies for combining the output of the two approaches (like aggregate parse probabilities and joint reranking) but we could not achieve any improvements over the *Replace* approach.

3.2 Reranker Features

The second step of our constituency pipeline is discriminative reranking. We conduct ranking experiments on the 50-best outputs of the product grammars. Like last year, we use a slightly modified version of the Mallet toolkit (McCallum, 2002), where the reranker is trained for the maximum entropy objective function of Charniak and Johnson (2005) and uses the standard feature set from Charniak and Johnson (2005) and Collins (2000) (**dflt**). This year we investigate new feature templates exploiting automatic dependency parses of the sentence in question (Farkas and Bohnet, 2012); Brown clusters (Brown et al., 1992); and atomic morphological feature values (Szántó and Farkas, 2014).

[6] As last year, we reduced compositional Korean POS tags to their first atomic tag, but since we could not apply the same method on Korean unlabeled data, we could not employ the *ExtendLex* approach on this language.

We create features from the full morphological description by using each morphological feature separately (**morph**). This approach allows us to combine a word with its morphological features (Dog-N-Cas=n). New features are established using the constituency labels and morphological features of the word's head, as well as morphological features of the head and its dependent. As we only use the main POS tags in case of the *Replace* method, these new features could only be applicable to *ExtendLex*.

We also created features based on automatic dependency parsing (**dep**). These features are made from heads of constituents and their dependency relations. We are using features describing relations between the same head-dependent pairs in both the constituency and dependency parses. The frequency of these relations is also used.

We generate Brown cluster-based features (**Brown**) using the already mentioned feature templates. In all features that contain words we replace words with their Brown cluster (to a pre-set depth). The Brown cluster features improve our results in *ExtendLex*, but have a negative effect in *Replace*.

Table 3 shows the reranking results on the development sets. Reranking with default features improves the scores over product grammars both for $ExtendLex$ and $Replace$ for all languages except Korean $Replace$, and Polish $ExtendLex$ and $Replace$. Adding new features to the reranker also results in higher scores across languages and methods with an exception of Swedish $ExtendLex$. The highest jumps (2.2% absolute) are achieved by adding **dep** features to the Polish $Replace\ Reranked_{dflt}$ grammar and by adding **morph+Brown+dep** features to the Basque $ExtendLex\ Reranked_{dflt}$ grammar.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
ExtendLex Reranked$_{dflt}$	81.59	81.92	82.83	90.16	91.06	-	89.79	79.09
Replace Reranked$_{dflt}$	86.11	82.30	84.59	90.02	91.09	83.50	88.31	78.87
ExtendLex Reranked$_{dflt+morph+Brown+dep}$	83.83	82.76	84.69	**90.76**	**92.05**	-	**91.44**	78.78
Replace Reranked$_{dflt+dep}$	**86.73**	**82.78**	**86.05**	90.47	91.89	**84.78**	90.53	**79.38**

Table 3: PARSEVAL scores of the reranker on the development sets for the predicted setting. dflt denotes default feature set of the reranker. Best results for each language are displayed in bold.

4 Dependency Parsing

Our dependency parsing architecture extends our 2013 system, as depicted in Figure 1. Like last year, it consists of two main steps: the parsing step and the ranking step. In the parsing step, we use several different dependency parsers to create a list of potential trees for each sentence, which are ranked in the ranking step to find the best analysis.

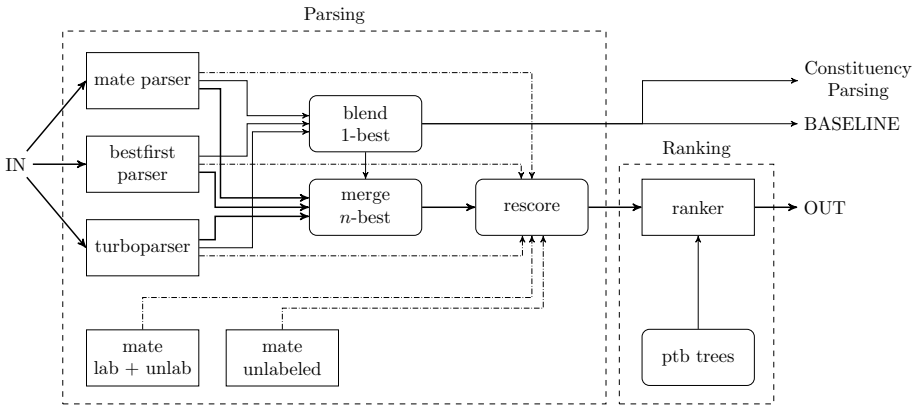

Figure 1: The architecture of the 2014 dependency parsing system.

4.1 The Parsing Step

We use three different dependency parsers to parse the data: the **mate parser**[7] (Bohnet, 2010), a second-order dependency parser with a Carreras-decoder (Carreras, 2007), the **BestFirst parser**, an in-house

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

implementation of the Easy-First parser (Goldberg and Elhadad, 2010), which we extended to handle non-projective structures similar to Tratz and Hovy (2011), and **TurboParser**,[8] which is based on linear programming relaxations (Martins et al., 2010). Additionally, we changed the feature sets of mate parser and TurboParser to include features from supertags.

The mate parser uses McDonald and Pereira's (2006) approximation algorithm to produce non-projective structures. It is a greedy hill-climbing process that starts from the highest-scoring projective tree and produces new trees by reattaching edges if they result in a higher overall score and do not violate the tree property. We use this mechanism in a non-greedy way to output n-best lists. The BestFirst parser uses beam search and thus can output n-best lists directly. The 50 highest-scoring trees from both parsers are merged into a list. This creates between 50 and 100 trees per sentence depending on the overlap between the source lists. This year, we extend the n-best list further in two ways: first, we add the 1-best tree produced by TurboParser. Second, we add a tree that is created by blending the 1-best trees from each of the three parsers. The blended trees constitute our baseline submission.

As the last step before ranking, all trees in the n-best lists are rescored by all parsing models. Last year, we found that the scores from each parsing model are very important features in the ranker. Since the n-best list for each sentence is created from multiple sources, not all of the trees in the list have been produced by all parsers and therefore do not necessarily have a score from each model, which makes rescoring necessary. At this point, we throw in two more parsing models, which were trained on automatically labeled data only or on a combination of automatically labeled data and treebanks. We thus get five different scores for each tree in each n-best list, which are then used as features in the ranker.

4.1.1 Supertags

Supertags (Bangalore and Joshi, 1999) are tags that encode more syntactic information than standard POS tags. Supertags have been used in deep grammar formalisms like CCG or HPSG to prune the search space for the parser. The idea has been applied to dependency parsing by Foth et al. (2006) and recently to statistical dependency parsing (Ouchi et al., 2014; Ambati et al., 2014) where supertags are used as features rather than to prune the search space.

Our supertag design originates from Ouchi et al. (2014). We tested various models with different levels of granularity and selected the three of them that give the most promising results. Models 1 and 2 are following Ouchi et al. (2014) and Model 3 is a modified version of Model 2 (see Table 4). None of the models encode the order among the dependents. Duplicate labels occurring on the same side of the word are removed.

	Description	Example (German)
Model 1	Relative position of the head of a word, its dependency relation and information whether the word's dependents occur to the right or to the left of the word.	OC/R+L
Model 2	Model 1 + dependency relations of obligatory dependents of verbs.	OC/R+L_OP/L
Model 3	Similar to Model 2; all dependency relations for all possible parts of speech, not only for verbs.	OC/R+MO/L_OP/L

Table 4: Models of supertags.

Table 5 gives supertag sizes in the training sets and its prediction accuracy with MarMoT. The number of tags increases and the tagging accuracy decreases with the amount of information encoded by the supertags. Due to the size of POS and label sets the number of tags strongly varies among languages, for instance the number of German supertags is five times larger than the Polish one in Model 3. Despite such differences and rather low accuracies of tagging (around 72% for Model 3) every model has a positive influence on the dependency parsing results.

	Basque		French		German		Hebrew		Hungarian		Korean		Polish		Swedish	
Model 1	80.59	(179)	82.86	(128)	85.08	(239)	76.84	(196)	79.44	(280)	90.72	(74)	81.66	(113)	72.41	(253)
Model 2	73.43	(1468)	80.41	(1179)	82.26	(1848)	74.75	(821)	77.01	(897)	90.82	(321)	78.84	(801)	69.66	(820)
Model 3	70.13	(4802)	74.42	(7609)	78.45	(14037)	68.30	(6924)	72.36	(10875)	76.21	(4196)	73.12	(2759)	64.41	(5386)

Table 5: MarMoT prediction accuracy and tag set size on the development sets for each supertag model.

[8] `http://www.ark.cs.cmu.edu/TurboParser`. We train TurboParser with MODELTYPE=FULL.

Supertags are used as features in both the mate parser and TurboParser. Their behavior differed when we experimented with different templates. In particular, the mate parser performed best when we used a considerably smaller set of templates compared to TurboParser. Table 6 lists the feature templates that we used in both parsers. We did not use supertags for the BestFirst parser because of time constraints.

both	s_h; $s_h w_h$; $s_h l_h$; $s_h p_h$; s_d; $s_d w_d$; $s_d l_d$; $s_h s_d$; $s_h p_d$; $s_h l_d$; $s_h w_d$; $s_d w_h$; $s_d l_h$
mate	$s_h w_h p_d$; $s_h w_d p_d$; $s_d w_d p_h$; $s_d w_h p_h$; $p_h s_h p_d$
turbo	$s_d p_d$; $s_d p_h$; s_{h+1}; s_{h-1}; $s_{h+1} w_{h+1}$; $s_{h-1} w_{h-1}$; $s_h s_{h+1}$; $s_h s_{h-1}$; s_{d+1}; s_{d-1}; $s_{d+1} w_{d+1}$; $s_{d-1} w_{d-1}$; $s_d s_{d+1}$; $s_d s_{d-1}$; s_{h+2}; s_{h-2}; $s_{h+2} w_{h+2}$; $s_{h-2} w_{h-2}$; $s_h s_{h+1} s_{h+2}$; $s_h s_{h-1} s_{h-2}$; s_{d+2}; s_{d-2}; $s_{d+2} w_{d+2}$; $s_{d-2} w_{d-2}$; $s_d s_{d+1} s_{d+2}$; $s_d s_{d-1} s_{d-2}$

Table 6: Feature definitions using supertags for the mate parser and TurboParser. s: supertag, w: word form, l: lemma, p: POS tag, h: head, d: dependent, $+1$: next token on the right, -1: next token on the left. In TurboParser, these features are also used for second order combinations, where head and dependent are replaced with dependent and sibling or grandparent and dependent, respectively.

Table 7 presents a comparison between parser results for all models. Supertags used with TurboParser yield improvements for all languages, giving an improvement of up to 1.5 points LAS (Polish, Swedish). We therefore decided to select the best model for each language individually for TurboParser. The only exception is German for which Model 3 gives the best results but has so many tags that learning time is not acceptable. Instead we use Model 2 which is negligibly worse.

The inclusion of supertags in the mate parser also gives some improvements but they are smaller than for TurboParser. The difference between all of the models is less than a 0.3 point LAS, so we decided to use Model 1 for all languages simply because it is the fastest one.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
TurboParser results								
Baseline	88.95/83.98	88.00/84.03	93.45/91.32	86.00/78.99	86.82/82.50	88.38/86.08	90.56/85.27	82.80/75.62
Model 1	89.52/85.08	88.22/84.43	93.66/91.69	86.44/79.64	87.33/83.34	**88.97/86.92**	91.52/86.29	83.42/77.01
Model 2	**89.64/85.08**	88.00/84.18	93.66/91.69	**86.69/80.05**	**87.44/83.39**	88.85/86.76	91.41/86.21	83.04/76.45
Model 3	89.55/84.79	**88.29/84.47**	**93.70/91.72**	86.35/79.74	87.42/**83.40**	88.86/86.70	**92.28/87.03**	**83.62/77.18**
mate results								
Baseline	88.78/83.96	87.95/84.34	93.04/91.25	86.59/79.66	87.89/84.15	87.82/85.49	91.47/85.96	82.96/76.50
Model 1	**89.29/84.74**	**88.42/84.79**	93.25/91.49	86.43/79.66	88.12/**84.47**	**88.65/86.52**	91.31/86.23	83.51/77.25
Model 2	88.93/84.56	88.31/84.67	**93.29/91.56**	**86.59/79.70**	**88.14**/84.42	88.52/86.38	91.00/85.97	**83.94/77.51**
Model 3	88.92/84.20	88.20/84.58	93.21/91.45	85.93/79.08	88.11/84.41	88.07/85.85	**91.58/86.46**	83.49/77.13

Table 7: UAS/LAS of parsers with different supertag models on the development sets for the predicted setting. Best results for each language are displayed in bold.

4.1.2 Blending

We implemented a parser **blender** (also known as a reparser; Sagae and Lavie, 2006). Blending is a simple way of combining parse trees from different parsers into one. Arcs are assigned scores depending on how frequent they are in the base parses. In the simplest case, each arc gets a score equal to the number of base parses it occurred in. Specifically, we implemented the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which is a directed maximum spanning tree algorithm. We applied this to combine the best parses from each of the individual parsers. If the blended tree was not already in the n-best list, it was added. Furthermore, the blended tree received a special flag, such that it can be used as a feature in the subsequent ranking step.

Table 8 shows the accuracies of each individual parser as well as the blended trees. The blended trees are better than any of the base parsers on several languages. In all cases where the blended tree is not the best, the BestFirst parser is lagging behind the mate parse and TurboParser quite considerably. This feeds into the blend and lowers the performance.[9]

[9]Sagae and Lavie (2006) additionally define more fine-grained blending schemes, where the contribution of arcs is weighted by which parser it originated from. We did not experiment with such schemes since the blended trees primarily are meant as a baseline, although we note that the figures in Table 8 suggest that weighting the base parsers probably would be beneficial.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
mate	89.29/84.74	88.42/84.78	93.25/91.49	86.43/79.66	**88.12/84.47**	88.65/86.52	91.32/86.23	83.51/77.25
bestfirst	83.72/75.76	86.88/83.33	92.73/90.91	85.23/78.60	83.30/75.52	86.41/83.75	89.66/82.52	82.55/75.78
turbo	**89.64/85.08**	88.29/84.47	93.66/91.69	86.69/80.05	87.44/83.39	**88.97/86.92**	**92.28/87.03**	83.62/77.18
blend	89.61/84.71	**88.75/85.10**	**93.93/92.19**	**87.46/80.65**	88.09/84.24	88.93/86.83	92.21/86.97	**84.72/78.23**
mate$_{ulbl}$	88.59/83.82	86.24/82.43	90.70/88.35	84.59/78.12	86.17/82.26	87.83/85.73	91.12/85.92	82.31/75.48
mate$_{lbl+ulbl}$	89.38/85.02	88.26/84.60	93.21/91.34	86.40/79.95	88.07/84.38	88.45/86.33	91.79/86.63	84.23/78.10

Table 8: UAS/LAS of each parser on the development sets for the predicted setting. Best results are shown in bold. The mate parser uses Model 1 supertags and TurboParser uses the best supertag model.

4.1.3 Parsing with Unlabeled Data

In order to make use of the unlabeled data, we trained two additional models on it using the mate parser. We first filtered the unlabeled data with a series of filters. We kept all sentences that fulfill the following constraints:

1. $5 \leq length \leq 20$
2. at most 2 unknown word forms wrt. the training data
3. contain at least one verb (determined by POS)
4. no word forms longer than 20 characters
5. no word forms that have more than 3 punctuation characters
6. no word forms that occur less than 5 times in the unlabeled data

These sentences were automatically annotated with lemmas, POS and morphological tags. We then parsed them with the mate parser and TurboParser (not using any supertags). The output of both parsers was then intersected following Sagae and Tsujii (2007) and filtered once more. This last filter removes sentences in which dependent labels that should occur only once per head occur more often (e.g., two subjects). We decided on the labels by their occurrence in the gold standard of the training data.

From such annotated unlabeled data we train two self-trained models (Charniak, 1997; McClosky et al., 2006). We sample two data sets for each language: one data set with 100k sentences and one with as many sentences as are in the training data. The first set is used to train a mate parser model on purely unlabeled data, the second set is combined with the treebank training data and we train a mate parser model on the combination. These self-trained models are then used as additional rescoring models in the parsing step (see Figure 1). The last two lines in Table 8 show the performance of these models on the development sets. The models trained on unlabeled data only usually lag behind quite a bit, but the models trained on a combination of labeled and unlabeled data are often not that far behind the canonical models. For Swedish, the model trained on the combination even surpasses all single-parser models, a curious result that deserves further investigation in the future.

4.2 The Ranking Step

The final step of our dependency parsing architecture is the ranking step. We use the same version of the Mallet toolkit (McCallum, 2002) that we use for constituency reranking to train the ranker (cf. Section 3). The ranking model is trained to select the best parse according to the labeled attachment score (LAS). We created training data for the ranker by running 5-fold cross-validation on the training sets using the base parsers (including blending and rescoring with all models).

We tuned the feature sets for each language individually by running greedy forward and backward feature selection on the training folds, starting from a baseline feature set. Features were added or removed based on improvements in the micro-averaged LAS across all folds. Most of the feature templates we use originate from our submission last year (Björkelund et al., 2013). Both the baseline and optimized feature sets are listed in Table 9. The corresponding abbreviations are described below.

default	B, M, T, GB, GM, GT, I	default	B, M, T, GB, GM, GT, I
Basque	B, M, T, GB, GM, GT, I, ptbp, *L*, FL, *GL*	Hungarian	B, M, T, GB, GM, GT, I, ptbp, *L*, FL, Blend
French	B, M, T, GB, GM, GT, I, ptbp, *L*, FL	Korean	B, M, T, GB, GM, GT, I, ptbp, *L*, FL
German	B, M, T, GB, GM, GT, I, ptbp, *L*, FL, BTProd	Polish	B, M, T, GB, GM, I, ptbp, *L*, FL, *GL*
Hebrew	B, M, T, GB, GM, GT, I, ptbp, *L*, FL, Blend	Swedish	B, M, T, GB, GM, GT, I, ptbp, *L*, FL, *U*

Table 9: Feature sets for the ranker for each language. Italics indicate features based on unlabeled data.

Scores from the base parsers – denoted **B**, **M**, and **T**, for the BestFirst, mate, and TurboParser, respectively. We also have additional indicator features that indicate whether a parse was the best according to each model, denoted **GB**, **GM**, and **GT**, respectively. Since the mate parser uses an approximate algorithm to handle non-projective dependencies, the greedy parse according to mate may not have the highest score. The GM feature is therefore a ternary feature, that indicates whether a parse was better, worse, or equal to the greedy parse. We additionally include the scores from the mate parser trained on unlabeled data and the mate parser trained on labeled and unlabeled data, denoted **U** and **L**, respectively, as well as the corresponding indicator feature for the parser trained on labeled and unlabeled data – denoted **GL**. Similar to last year we also experimented with non-linear combinations of the scores from various parsers. The only such feature that had a clear positive effect was the product of the scores from the BestFirst and TurboParser, denoted **BTProd**.

Blend features – indicator features that mark whether a tree was the output of the parser blender or not. This template additionally includes conjunctions that pair it with the indicator features from the other parsers (e.g., whether it was the blend and the best parse according to TurboParser).

Constituency features extracted from the constituency trees. Specifically, for every head-dependent pair, we extract the path in the constituency tree between the nodes, denoted **ptbp**.

Function label uniqueness – marks the number of dependents of a head with a certain label that should typically only occur once. We extracted lists on the training sets of labels that never occur more than once as the dependent of a word (e.g., subjects). Features are then extracted on heads that have one or multiple dependents with any of the labels in this list. This feature template is denoted **FL**.

The performance of the ranker on the development sets are shown in Table 10. The table also shows the baseline (i.e., the blended trees) as well as the oracle scores. Three feature sets are evaluated – the default feature set (*Ranked*$_{dflt}$), the optimized feature set (*Ranked*$_{opt}$), and the optimized feature set but without features that are based on unlabeled data (*Ranked*$_{no\text{-}ulbl}$). The table shows that ranking provides a strong boost over the baseline, ranging from about 0.5 (German) to 1.7 (Basque) in LAS. The contribution of the features based on unlabeled data is extremely modest.[10] Interestingly, however, we do see a considerable contribution from the features that are based on unlabeled data for Swedish. This is in line with the performance of these parsers on the development set (cf., Table 8) and suggests that if the self-trained parsers are good enough, they can make a valuable contribution.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Baseline	89.61/84.71	88.75/85.10	93.93/92.19	87.46/80.65	88.09/84.24	88.93/86.83	92.21/86.97	84.72/78.23
Ranked$_{dflt}$	90.22/85.80	88.68/85.00	93.92/92.19	87.21/80.49	88.12/84.34	89.52/87.41	92.70/87.59	84.37/77.98
Ranked$_{no\text{-}ulbl}$	90.61/86.40	**89.36**/86.00	94.23/92.72	88.55/**81.96**	88.67/84.99	**89.96/87.89**	93.13/88.00	85.38/79.02
Ranked$_{opt}$	**90.66/86.46**	89.35/**86.01**	**94.28/92.75**	**88.58**/81.93	**88.72/85.08**	89.91/87.85	**93.19/88.07**	**85.73/79.64**
Oracle	95.02/91.66	92.97/90.31	97.95/97.15	92.37/87.07	91.91/88.37	96.33/94.72	97.36/95.30	90.02/85.40

Table 10: UAS/LAS of the ranker on the development sets for the predicted setting. Baseline denotes the blended trees. The highest number in each column is indicated in bold.

5 Test Set Results

We apply our systems on the test sets and present our scores in this section. For both tracks we also compare our results with the baselines provided by the organizers and with the best ST competitor.

The constituency parsing results are given in Table 11. The *Replace* system outperforms the *ExtendLex* system across all languages with the only exception of 0.01 difference in French. We have the highest accuracies for all languages except Polish.

Table 12 displays our results in the dependency track. We achieve the best accuracies for all languages. Our baseline system, i.e., blended trees, comes third for Hungarian, and second for all other languages. Ranking systematically helps, up to 1.7% absolute LAS in Basque.

[10] Sometimes *Ranked*$_{no\text{-}ulbl}$ even does slightly better than *Ranked*$_{opt}$. We remind the reader that the feature sets were tuned on cross-validation over the training set. Hence such deviations are to be expected, especially when the contribution of these features is as minor as observed here.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
ST Baseline	74.74	80.38	78.30	86.96	85.22	78.56	86.75	80.64
Other	85.35	79.68	77.15	86.19	87.51	79.50	**91.60**	82.72
ExtendLex Reranked	83.78	**82.53**	79.76	89.75	90.76	-	89.19	82.94
Replace Reranked	**88.24**	82.52	**81.66**	**89.80**	**91.72**	**83.81**	90.50	**85.50**

Table 11: Final PARSEVAL F_1 scores for constituents on the test sets for the predicted setting. ST Baseline denotes the best baseline provided by the ST organizers. Other denotes the best competitor.

	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
ST Baseline	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	88.58/83.46	88.02/84.51	90.80/88.66	82.79/75.55	88.76/84.90	85.33/81.88	88.42/80.13	87.26/81.23
Baseline	88.76/83.97	88.45/84.83	91.03/88.62	86.63/80.77	88.79/84.51	88.37/86.42	90.93/86.21	87.72/81.42
Ranked	**89.96/85.70**	**89.02/85.66**	**91.63/89.58**	**87.41/81.65**	**89.57/85.59**	**89.10/87.27**	**91.48/86.75**	**88.48/82.75**

Table 12: Final UAS/LAS scores for dependencies on the test sets for the predicted setting. ST Baseline denotes the MaltParser baseline provided by the ST organizers (Ballesteros, 2013). Other denotes the best competitor.

6 Conclusion

We discuss our contribution to this year's Shared Task from two perspectives: How well we did as compared to last year and how we made use of unlabeled data.

Comparison to last year. While our entry for this year was generally the strongest among all participants, it should be noted that we did not entirely outperform our own submission from last year (Björkelund et al., 2013).[11] Broadly, this difference can be attributed to the lower quality of POS tags and morphological predictions compared to last year, as described in Section 2. This affected our models for the constituency and the dependency track.

For dependency parsing, the introduction of supertags could compensate the effect to some extent.

For constituency parsing, our strongest system (*Replace*) also relies on predictions from the POS/morphological tagger in its input, whose lower quality was partly compensated by a richer feature set in the ranker.

Utility of unlabeled data. One objective of this year's shared task was to leverage unlabeled data in the parsing systems. This appears to have been a rather tough challenge, judging both from our own experience as well as other participants.[12] While unlabeled data has been shown to be helpful for domain adaptation (Bacchiani et al., 2006; Blitzer et al., 2006; McClosky and Charniak, 2008), it should be noted that the experimental setting in the shared task is *not* a domain adaptation setting.[13] On the contrary, our training and test sets are drawn from the same domain, while the unlabeled data typically originate from some other domain.

Besides the inclusion of parser scores from parsers trained on unlabeled or a combination of labeled and unlabeled data in the dependency ranker, and the use of Brown cluster features in the constituency reranker, we had no success exploiting the unlabeled data. The extended lexicons in constituency parsing did help, but overall could not beat the *Replace* method. Initial experiments on using collocations extracted from the unlabeled data in the (re)rankers did not bring any improvements, although time prevented us from exploring this path conclusively.

In the dependency track, we saw extremely modest improvements from including scores drawn from self-trained parsers. A peculiar exception was the case of Swedish, where these features improve the scores of the ranker by more than half a point. This improvement aligns well with the surprisingly good self-trained parser for Swedish. Further research is required to understand the reasons for this.

[11] We achieved better constituency scores for Basque, German, Polish, and Korean; and better dependency scores for Basque, French, Hebrew, Korean, and Swedish. We refer the reader to our paper from last year for a detailed comparison.

[12] p.c., Djamé Seddah.

[13] i.e., an annotated data set from one domain, a test set from another, and unlabeled data from the same domain as the test set.

Acknowledgements

Agnieszka Faleńska is funded by the Project International computer science and applied mathematics for business study program at the University of Wrocław co-financed with EU funds within the European Social Fund (POKL.04.01.01-00-005/13). Richárd Farkas and Zsolt Szántó are funded by the EU and the European Social Fund through the project FuturICT.hu (TÁMOP-4.2.2.C-11/1/KONV-2012-0013). Thomas Müller is supported by a Google Europe Fellowship in NLP. 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: 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).

We also would like to thank the following institution for providing the unlabeled data set: The LDC for the Arabic Gigaword, the IXA NLP Research Group and Elhuyar Fundazioa for the Elhuyar Corpus Version, the French CNRTL for the Est Republicain Corpus, the IMS group and the university of Heidelberg for the German Wikidump, the Bar Ilan University for the Hebrew Wikidump, the University of Szeged for their Hungarian Newsire corpus, the KAIST for their Large Scale Korean Corpus, the Polish Institute of Science for their Wikipedia corpus, and the Språkbanken group at the University of Gothenburg for their Parole corpus.

Finally, we appreciate all the hard work from the organizers of the Shared Task, in particular Djamé Seddah.

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.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2014. Improving dependency parsers using combinatory categorial grammar. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 159–163, Gothenburg, Sweden, April. Association for Computational Linguistics.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech & Language*, 20(1):41 – 68.

Miguel Ballesteros. 2013. Effective morphological feature selection with MaltOptimizer at the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 53–60, Seattle, WA.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Anders Björkelund, Özlem Çetinoğlu, Richárd Farkas, Thomas Müller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

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.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

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.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pages 598–603. AAAI Press.

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. *CoRR*, abs/1309.1649.

Yoeng-jin Chu and Tseng-hong Liu. 1965. On the shortest aborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *COMPUTATIONAL LINGUISTICS*, 33.

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.

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

Richárd Farkas and Bernd Bohnet. 2012. Stacking of dependency and phrase structure parsers. In *Proceedings of COLING 2012*, pages 849–866, Mumbai, India, December. The COLING 2012 Organizing Committee.

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*.

Kilian A. Foth, Tomas By, and Wolfgang Menzel. 2006. Guiding a constraint dependency parser with supertags. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 289–296, Sydney, Australia, July. Association for Computational Linguistics.

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 and Michael Elhadad. 2013. Word Segmentation, Unknown-word Resolution, and Morphological Agreement in a Hebrew Parsing System. *Computational Linguistics*, 39(1):121–160.

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

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.

David McClosky and Eugene Charniak. 2008. Self-training for biomedical parsing. In *Proceedings of ACL-08: HLT, Short Papers*, pages 101–104, Columbus, Ohio, June. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.

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.

Hiroki Ouchi, Kevin Duh, and Yuji Matsumoto. 2014. Improving dependency parsers with supertags. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 154–158, Gothenburg, Sweden, April. 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.

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, New York City, USA, June. Association for Computational Linguistics.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050, Prague, Czech Republic, June. Association for Computational Linguistics.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. Smor: A german computational morphology covering derivation, composition, and inflection. In *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004*, pages 1263–1266.

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

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho Choi, Matthieu Constant, Richárd Farkas, 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, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2014. Overview of the SPMRL 2014 shared task on parsing morphologically rich languages. In *Notes of the SPMRL 2014 Shared Task on Parsing Morphologically-Rich Languages*, Dublin, Ireland.

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).

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.

Zsolt Szántó and Richárd Farkas. 2014. Special techniques for constituent parsing of morphologically rich languages. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 135–144, Gothenburg, Sweden, April. Association for Computational Linguistics.

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. 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*.

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

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*.