Effective Morphological Feature Selection with MaltOptimizer at the SPMRL 2013 Shared Task

Miguel Ballesteros
Natural Language Processing Group
Pompeu Fabra University.
Barcelona, Spain
miguel.ballesteros@upf.edu

Abstract

The inclusion of morphological features provides very useful information that helps to enhance the results when parsing morphologically rich languages. MaltOptimizer is a tool, that given a data set, searches for the optimal parameters, parsing algorithm and optimal feature set achieving the best results that it can find for parsers trained with MaltParser. In this paper, we present an extension of MaltOptimizer that explores, one by one and in combination, the features that are geared towards morphology. From our experiments in the context of the Shared Task on Parsing Morphologically Rich Languages, we extract an in-depth study that shows which features are actually useful for transition-based parsing and we provide competitive results, in a fast and simple way.

1 Introduction

Since the CoNLL Shared Tasks on Syntactic Dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007), the number of treebanks and new parsing methods have considerably increased. Thanks to that, it has been observed that parsing morphologically rich languages (henceforth, MRLs) is a challenge because these languages include multiple levels of information that are difficult to classify and, therefore, to parse. This is why there has been recent research in this direction, with for instance a Special Issue in Computational Linguistics (Tsarfaty et al., 2012b).

MaltOptimizer (Ballesteros and Nivre, 2012b; Ballesteros and Nivre, 2012a) is a system that is capable of providing optimal settings for training models with MaltParser (Nivre et al., 2006a), a freely available transition-based parser generator. MaltOptimizer, among other things, performs an in-depth feature selection, selecting the attributes that help to achieve better parsing results. In this paper – and in this participation in the Shared Task on Parsing Morphologically Rich Languages (Seddah et al., 2013) – we present an extension of MaltOptimizer that performs a deeper search over the morphological features that are somewhat one of the keys to parsing MRLs. Instead of lumping all morphosyntactic features together, we define a different field for each individual feature (case, number, gender, etc.). Hence, we are able to extract a study that shows which features are actually useful for parsing MRLs with MaltParser.

The new *SPMRL-MaltOptimizer* implementation is available for download at *http://nil.fdi.ucm.es/maltoptimizer/spmrl.html*.
It is worth noting that it can be applied to any treebank in CoNLL data format.[1]

The rest of the paper is organized as follows. Section 2 describes MaltOptimizer. Section 3 shows how we modified MaltOptimizer to make it able to perform a more complete morphological feature selection. Section 4 describes the experiments that we carried out with the data sets of the Shared Task on Parsing Morphologically Rich Languages. Section 5 reports the results of the experiments and the conclusions that we can extract. Section 6 discusses related work on MaltOptimizer and parsing morphologically rich languages. And finally, Section 7 con-

[1] http://ilk.uvt.nl/conll/#dataformat

Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically Rich Languages, pages 63–70,
Seattle, Washington, USA, 18 October 2013.

cludes.

2 MaltOptimizer

MaltOptimizer is a system written in Java that implements a full optimization procedure for MaltParser based on the experience acquired from previous experiments (Hall et al., 2007; Nivre and Hall, 2010). MaltOptimizer attempts to find the best model that it can find, but it does not guarantee that the outcome is the best model possible because of the difficulty of exploring all the possibilities that are provided by the parameters, parsing algorithms and different feature windows. The optimization procedure is divided in 3 different phases, as follows:

1. Data analysis and initial optimization.
2. Parsing algorithm selection.
3. Feature selection and LIBLINEAR optimization.

MaltOptimizer divides the treebank into a training set and a held-out test set for evaluation. In the first phase, MaltOptimizer makes an analysis of the treebank in order to set up the rest of the optimization, and it attempts the optimization with some general parameters, such as the way of handling covered roots.[2] After that, it tests the parsing algorithms that are available in MaltParser by selecting the one that provides best results in default settings. In the third phase, it explores a wide range of features that are based on previous parsing steps and/or the information annotated in the treebanks. Finally, it also explores the single hyper-parameter (c) of the LIBLINEAR classifier.

In the next Section, we present how we updated MaltOptimizer for our participation in the Shared Task of parsing MRLs.

3 Morphological Feature Exploration

The CoNLL data format contains several columns of information that help to perform the dependency parsing of a sentence. One of the columns is the FEATS column that normally contains a set of morphological features, which is normally of the format a=x|b=y|c=z. At the time of writing, the available version of MaltOptimizer explores the features included in this column as a single feature, by lumping all morphosyntactic features in the MaltParser classifier, and by splitting the information but including all of them at the same time without making any distinctions. This is what MaltParser allows by using the standard CoNLL format, which contains the following information per column.

1. ID: Identifier.
2. FORM: Word form.
3. LEMMA: Lemma or stemmed version of the word.
4. CPOSTAG: Coarse-grained part-of-speech tag.
5. POSTAG: Fine-grained part-of-speech tag.
6. FEATS: Morphosyntactic features (e.g., case, number, tense, etc.). It is normally of the format a=x|b=y|c=z.
7. HEAD: Head node.
8. DEPREL: Dependency relation to head.
9. PHEAD: Projective head node.
10. PDEPREL: Projective dependency relation to head.

However, MaltParser also provides the option of parsing new data formats that are derived from the original CoNLL format. Therefore, there is the possibility to add new columns that may contain useful information for parsing. The new MaltOptimizer implementation automatically generates a new data format and a new data set. It creates new columns that only contain the information of a single feature which is included in the FEATS column.

Figure 1 shows two versions of a sentence annotated in the French treebank from the Shared Task. The one shown above is in the standard CoNLL format, and the one shown below is the extended format generated by MaltOptimizer in which the FEATS column has been divided in 10 different columns.

[2] A covered root is a root node covered by a dependency arc.

```
1  En      en      P       P       mwehead=ADV+|pred=y     4       mod
2  tout    tout    D       DET     g=m|n=s|s=ind|pred=y    1       dep_cpd
3  cas     cas     N       NC      g=m|s=c|pred=y  1       dep_cpd
4  est     łtre    V       V       m=ind|n=s|p=3|t=pst     0       root
5  -il     il      CL      CLS     g=m|n=s|p=3|s=suj       4       suj
6  plus    plus    ADV     ADV     _       7       mod
7  nuanc   nuanc   A       ADJ     g=m|n=s|s=qual  4       ats
8  .       .       PONCT   PONCT   s=s     4       ponct


1  En      en      P       P       _       _       _       _       _       _       ADV+    y       _       _       4   mod
2  tout    tout    D       DET     ind     m       s       _       _       _       _       y       _       _       1   dep_cpd
3  cas     cas     N       NC      c       m       _       _       _       _       _       y       _       _       1   dep_cpd
4  est     łtre    V       V       _       _       s       3       ind     pst     _       _       _       _       0    root
5  -il     il      CL      CLS     suj     m       s       3       _       _       _       _       _       _       4    suj
6  plus    plus    ADV     ADV     _       _       _       _       _       _       _       _       _       _       7    mod
7  nuanc   nuanc   A       ADJ     qual    m       s       _       _       _       _       _       _       _   4   ats
8  .       .       PONCT   PONCT   s       _       _       _       _       _       _       _       _       _       4    ponct
```

Figure 1: A sentence from the French treebank in the standard (above) and complex (below) formats. The projective columns have been removed for simplicity.

4 Experiments

With the intention of both assessing the usefulness of the new MaltOptimizer implementation and testing which features are useful for each targeted language, we carried out a series of experiments over the data sets from the Shared Task on Parsing MRLs (Seddah et al., 2013). We run the new MaltOptimizer implementation for all the data sets provided by the Shared Task organizers and we run MaltParser with the model suggested. Therefore, we had 36 different runs, 4 for each language (*gold* and *predicted* scenarios with *5k* treebanks, and *gold* and *predicted* scenarios with *full* treebanks).

In order to have a comparable set of results, we performed all the optimization processes with the smaller versions of the treebanks (*5k*) and both optimization and training steps with both the small and larger version for all languages. Each MaltOptimizer run took approximately 3-4 hours for optimization (the running time also depends on the size of the set of morphological features, or other parameters, such as the number of dependency relations) and it takes around 20 extra minutes to get the final model with MaltParser. These estimates are given with an Intel Xeon server with 8 cores, 2.8GHz and a heap space of, at least, 8GB.

5 Results and Discussion

Table 1 shows the results for *gold*-standard input while Table 2 shows the results for the provided *predicted* inputs for the best model that the new MaltOptimizer implementation can find (Dev-5k, Dev, Test-5k and Test) and a baseline, which is MaltParser in default settings (Malt-5k and Malt) on the test sets. The first conclusion to draw is that the difference between *gold* and *predicted* inputs is normally of 2 points, however for some languages such as French the drop reaches 6 points. It is also evidenced that, as shown by Ballesteros and Nivre (2012a), some languages benefit more from the feature selection phase, while others achieve higher improvements by selecting a different parsing algorithm.

In general terms, almost all languages benefit from having an accurate stemmed version of the word in the LEMMA column, providing very substantial improvements when accurately selecting this feature. Another key feature, for almost all languages, is the grammatical CASE that definitely enhances the performance; we can therefore conclude that it is essential for MRLs. Both aspects evidence the lexical challenge of parsing MRLs without using this information.

There is a positive average difference comparing with the MaltParser baseline of 4.0 points training over the full treebanks and predicted scenario and 5.6 points training over the full treebanks and gold scenario. It is therefore evident how useful MaltOptimizer is when it can perform an in-depth morphological feature exploration. In the following subsections we explain the results for each targeted language, giving special emphasis to the ones that turn out to be more meaningful.

5.1 Arabic

For Arabic, we used the shared task Arabic data set, originally provided by the LDC (Maamouri et

Language	Default	Phase 1	Phase 2	Phase 3	Diff	Dev-5k	Dev	Malt-5k	Malt	Test-5k	Test
Arabic	83.48	83.49	83.49	87.95	4.47	85.98	87.60	80.36	82.28	85.30	87.03
Basque	67.05	67.33	67.45	79.89	13.30	80.35	81.65	67.13	69.19	81.40	82.07
French	77.96	77.96	78.27	85.24	7.28	85.19	86.30	78.16	79.86	84.93	85.71
German	79.90	81.09	84.85	87.70	7.80	87.32	90.40	76.64	79.98	83.59	86.96
Hebrew	76.78	76.80	79.37	80.17	3.39	79.83	79.83	76.61	76.61	80.03	80.03
Hungarian	70.37	71.11	71.98	81.91	11.54	80.69	80.74	71.27	72.34	82.37	83.14
Korean	87.22	87.22	87.22	88.94	1.72	86.52	90.20	81.69	88.43	83.74	89.39
Polish	75.52	75.58	79.28	80.27	4.75	81.58	81.91	76.64	77.70	79.79	80.49
Swedish	76.75	76.75	78.91	79.76	3.01	74.85	74.85	75.73	75.73	77.67	77.67

Table 1: Labeled attachment score per phase compared to default settings for all training sets from the Shared Task on PMRLs in the *gold* scenario on the held-out test set for optimization. The first columns shows results per phase (the procedure of each phase is briefly described in Section 2) on the held-out sets for evaluation. The **Dev-5k** and **Dev** columns report labeled attachment score on the development sets. The columns **Malt** and **Malt-5k** report results of MaltParser in default settings on the test sets. And the columns, **Test-5k** and **Test** report results for the best model found by SPMRL-MaltOptimizer on the test sets.

Language	Default	Phase 1	Phase 2	Phase 3	Diff	Dev-5k	Dev	Malt-5k	Malt	Test-5k	Test
Arabic	83.20	83.21	83.21	85.68	2.48	80.35	82.28	78.30	80.36	79.64	81.90
Basque	68.80	69.33	69.89	77.24	8.44	78.12	79.46	68.12	70.11	77.59	78.58
French	77.43	77.43	77.63	79.42	1.99	77.65	79.33	76.54	77.98	77.56	79.00
German	78.69	79.87	82.58	83.97	5.28	83.39	86.63	74.81	77.81	79.22	82.75
Hebrew	76.29	76.31	79.01	79.67	3.38	73.40	73.40	69.97	69.97	73.01	73.01
Hungarian	68.26	69.12	69.96	78.71	10.45	76.82	77.62	69.08	70.15	79.00	79.63
Korean	80.08	80.08	80.08	81.63	1.55	77.96	83.02	74.87	82.06	75.90	82.65
Polish	74.43	74.49	76.93	78.41	3.98	80.61	80.83	75.29	75.63	79.50	80.49
Swedish	74.53	74.53	76.51	77.66	3.13	72.90	72.90	73.21	73.21	75.82	75.82

Table 2: Labeled attachment score per phase compared to default settings for all training sets from the Shared Task on PMRLs in the *predicted* scenario on the held-out test set for optimization. The columns of this table report the results in the same way as Table 1 but using predicted inputs.

al., 2004), specifically its SPMRL 2013 dependency instance, derived from the Columbia Catib Treebank (Habash and Roth, 2009; Habash et al., 2009), extended according to the SPMRL 2013 extension scheme (Seddah et al., 2013).

For the *gold* input, the most useful feature is, by far, DASHTAG[3] with an improvement of 2 points. CASE is also very useful, as it is for most of the languages, with 0.67 points. Moreover, SUBCAT (0.159) and CAT (0.129) provide improvements as well.

In the *pred* scenario, there is no DASHTAG, and this allows other features to rise, for instance, CASE (0.66), CPOSTAG (0.12), GENDER (0.08), SUBCAT (0.07) and CAT (0.06) provide improvements. Finally it is worth noting that the TED accuracy (Tsarfaty et al., 2011) for the lattices is 0.8674 with the full treebanks and 0.8563 with 5k treebanks, which overcomes the baseline in more than 0.06 points, this shows that MaltOptimizer is also useful under TED evaluation constraints.

[3]DASHTAG comes from the original constituent data, when a DASHTAG was present in a head node label, this feature was kept in the Catib corpus.

5.2 Basque

The improvement provided by the feature selection for Basque (Aduriz et al., 2003) is really high. It achieves almost 13 points improvement with the *gold* input and around 8 points with the *predicted* input. The results in the gold scenario are actually a record if we also consider the experiments performed over the treebanks of the CoNLL Shared Tasks (Ballesteros and Nivre, 2012a). One of the reasons is the treatment of covered roots that is optimized during the first phase of optimization. This corpus has multiple root labels, ROOT being the most common one and the one selected by MaltOp-

timizer as default.

For the *gold* input, the CPOSTAG and LEMMA columns turn out to be very useful, providing an improvement of 2.5 points and slightly less than 1 point respectively, MaltOptimizer selects them all over the more central tokens over the stack and the buffer. The Basque treebank contains a very big set of possible features in the FEATS column, however only some of them provide significant improvements, which evidences the usefulness of selecting them one by one. The most useful feature with a huge difference is *KASE* (or CASE) that provides 5.9 points by itself. MaltOptimizer fills out all the available positions of the stack and the buffer with this feature. Another useful feature is ERL [type of subordinated sentence], providing almost 0.8 points. Moreover, NUMBER (0.3), NORK2 (0.15), ASP [aspect] (0.09), NOR1 (0.08), and NMG (0.06) provide slighter, but significant, improvements as well.[4]

Surprisingly, the *predicted* input provides better results in the first 2 phases, which means that for some reason MaltParser is able to parse better by using just the predicted POS column, however, the improvement achieved by MaltOptimizer during Phase 3 are (just) a bit more than 7 points. In this case, the CPOSTAG column is less useful, providing only 0.13 points, while the LEMMA (1.2) is still very useful. CASE provides 4.5 points, while NUM (0.17), ASP (0.13) and ADM (0.11) provide improvements as well.

5.3 French

For French (Abeillé et al., 2003) there is a huge difference between the results with *gold* input and the results with *predicted* input. With *gold* input, the feature selection provides a bit less than 8 points while there is just an improvement of around 2 points with *predicted* input. In this case, the lack of quality in the predicted features is evident. It is also interesting that the lexical column, FORM, provides a quite substantial improvement when MaltOptimizer attempts to modify it, which is something that does not happen with the rest of languages.

For the *gold* input, apart from LEMMA that provides around 0.7 points, the most useful feature is MWEHEAD [head of a multi word expression, if exists] that does not exist in the *predicted* scenario. MWEHEAD provides more than 4 points; this fact invites us to think that a predicted version of this feature would be very useful for French, if possible. PRED [automatically predicted] (0.8), G [gender] (0.6), N [number] (0.2) and S [subcat] (0.14) are also useful.

In the *predicted* scenario, the CPOSTAG column provides some improvements (around 0.1) while the LEMMA is less useful than the one in the gold scenario (0.2). The morphological features that are useful are S [subcat] (0.3) and G [gender] (0.3).

5.4 German

For German (Brants et al., 2002) the results are more or less in the average. For the *gold* input, LEMMA is the best feature providing around 0.8 points; from the morphological features the most useful one is, as expected, CASE with 0.58 points. GENDER (0.16) and NUMBER (0.16) are also useful.

In the *predicted* scenario, CASE is again very useful (0.67). Other features, such as, NUMBER (0.10) and PERSON (0.10) provide improvements, but as we can observe a little bit less than the improvements provided in the *gold* scenario.

5.5 Hebrew

For the Hebrew (Sima'an et al., 2001; Tsarfaty, 2013) treebank, unfortunately we did not see a lot of improvements by adding the morphological features. For the *gold* input, only CPOSTAG (0.08) shows some improvements, while the *predicted* scenario shows improvements for NUM (0.08) and PER (0.08). It is worth noting that the TED accuracy (Tsarfaty et al., 2011) for the lattices is 0.8305 which is ranked second.

This outcome is different from the one obtained by Goldberg and Elhadad (2010), but it is also true that perhaps by selecting a different parsing algorithm it may turn out different, because two parsers may need different features, as shown by Zhang and Nivre (2012). This is why, it would be very interesting to perform new experiments with MaltOptimizer by testing different parsing algorithms included in MaltParser with the Hebrew treebank.

[4] NORK2, NOR1 and NMG are auxiliaries case markers.

5.6 Hungarian

The Hungarian (Vincze et al., 2010) results are also very consistent. During the feature selection phase, MaltOptimizer achieves an improvement of 10 points by the inclusion of morphological features. This also happens in the initial experiments performed with MaltOptimizer (Ballesteros and Nivre, 2012a), by using the Hungarian treebank of the CoNLL 2007 Shared Task. The current Hungarian treebank presents covered roots and multiple root labels and this is why we also get substantial improvements during Phase 1.

For the *gold* input, as expected the LEMMA column is very useful, providing more than 1.4 points, while MaltOptimizer selects it all over the available feature windows. The best morphological feature is again CASE providing an improvement of 5.7 points just by itself, in a similar way as in the experiments with Basque. In this case, the SUBPOS [grammatical subcategory] feature that is included in the FEATS column is also very useful, providing around 1.2 points. Other features that are useful are NUMP [number of the head] (0.2), NUM [number of the current token] (0.16), DEF [definiteness] (0.11) and DEG [degree] (0.09).

In the *predicted* scenario, we can observe a similar behavior for all features. MOOD provides 0.4 points while it does not provide improvements in the *gold* scenario. The results of the SUBPOS feature are a bit lower in this case (0.5 points), which evidences the quality lost by using *predicted* inputs.

5.7 Korean

As Korean (Choi, 2013) is the language in which our submission provided the best results comparing to other submissions, it is interesting to dedicate a section by showing its results. For the *5k* input, our model provides the best results of the Shared Task, while the results of the model trained over the *full* treebank qualified the second.

For the *gold* input, the most useful feature is CPOSTAG providing around 0.6 points. Looking into the morphological features, CASE, as usual, is the best feature with 0.24 points, AUX-Type (0.11), FNOUN-Type (0.08) are also useful.

In the *predicted* scenario, MaltOptimizer performs similarly, having CPOSTAG (0.35) and CASE (0.32) as most useful features. ADJ-Type (0.11) and PUNCT-Type (0.06) are also useful. The results of the features are a bit lower with the *predicted* input, with the exception of CASE which is better.

5.8 Polish

Polish (Świdziński and Woliński, 2010) is one of the two languages (with Swedish) in which our model performs with the worst results.

In the *gold* scenario only the LEMMA (0.76) shows some substantial improvements during the optimization process; unfortunately, the morphological features that are extracted when MaltOptimizer generates the new complex data format did not fire.

For the *predicted* input, LEMMA (0.66) is again the most useful feature, but as happened in the *gold* scenario, the rest of the features did not fire during the feature selection.

5.9 Swedish

As happened with Polish, the results for Swedish (Nivre et al., 2006b) are not as good as we could expect; however we believe that the information shown in this paper is useful because MaltOptimizer detects which features are able to outperform the best model found so far and the model trained with MaltParser in default settings by a bit less than 2 points in the predicted scenario and more than 2 points in the gold scenario.

For the *gold* scenario only two features are actually useful according to MaltOptimizer, MaltOptimizer shows improvements by adding GENDER (0.22) and PERFECTFORM (0.05).

For the *predicted* input, MaltOptimizer shows improvements by adding DEGREE (0.09), GENDER (0.08) and ABBRV (0.06). However, as we can see the improvements for Swedish are actually lower compared to the rest of languages.

6 Related Work

There has been some recent research making use of MaltOptimizer. For instance, Seraji et al. (2012) used MaltOptimizer to get optimal models for parsing Persian. Tsarfaty et al. (2012a) worked with MaltOptimizer and Hebrew by including the optimization for presenting new ways of evaluating statistical parsers. Mambrini and Passarotti (2012),

Agirre et al. (2012), Padró et al. (2013) and Ballesteros et al. (2013) applied MaltOptimizer to test different features of Ancient Greek, Basque and Spanish (the last 2) respectively; however at that time MaltOptimizer did not allow the FEATS column to be divided. Finally, Ballesteros et al. (2012) applied MaltOptimizer for different parsing algorithms that are not included in the downloadable version showing that it is also possible to optimize different parsing algorithms.

7 Conclusions

This new MaltOptimizer implementation helps the developers to adapt MaltParser models to new languages in which there is a rich set of features. It shows which features are able to make a change in the parsing results and which ones are not, in this way, it is possible to focus annotation effort for the purpose of parsing. We clearly observe that MaltOptimizer outperforms very substantially the results shown in the baseline, which is MaltParser in default settings, and it is also nice to see that the improvements provided by MaltOptimizer for the morphological features are actually very high, if we compare to the ones obtained by MaltOptimizer for the corpora of the CoNLL shared tasks (Ballesteros and Nivre, 2012a).

It is worth noting that the experiments with MaltOptimizer do not take so long. The time needed to perform the optimization is actually very short if we compare to the efforts needed to achieve results in the same range of accuracy by careful manual optimization. The MaltOptimizer process was sped up following heuristics derived from deep proven experience (Nivre and Hall, 2010), which means that there are several combinations that are untested; however, it is worth noting that these heuristics resulted in similar performance to more exhaustive search for a big set of languages (Ballesteros, 2013).

From the feature study shown in Section 5, we expect that it could be useful for people doing parsing research and interested in parsing MRLs. Finally, comparing our submission with the results of other teams, we believe that we provide a fast and effective parser optimization for parsing MRLs, having competitive results for most of the languages.

Acknowledgments

I would like to thank Koldo Gojenola who initially gave me the idea presented in this paper. I am also very thankful to Joakim Nivre for his constant help and support. Finally, special thanks to the organizers Djamé Seddah, Reut Tsarfaty and Sandra Kübler.

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 *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

Eneko Agirre, Aitziber Atutxa, and Kepa Sarasola. 2012. Contribution of complex lexical information to solve syntactic ambiguity in Basque. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, India, 12/2012.

Miguel Ballesteros and Joakim Nivre. 2012a. MaltOptimizer: A System for MaltParser Optimization. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*.

Miguel Ballesteros and Joakim Nivre. 2012b. MaltOptimizer: An Optimization Tool for MaltParser. In *Proceedings of the System Demonstration Session of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*.

Miguel Ballesteros, Carlos Gómez-Rodríguez, and Joakim Nivre. 2012. Optimizing Planar and 2-Planar Parsers with MaltOptimizer. *Procesamiento del Lenguaje Natural*, 49, 09/2012.

Miguel Ballesteros, Simon Mille, and Alicia Burga. 2013. Exploring Morphosyntactic Annotation Over a Spanish Corpus for Dependency Parsing . In *Proceedings of the Second International Conference on Dependency Linguistics (DEPLING 2013)*.

Miguel Ballesteros. 2013. Exploring Automatic Feature Selection for Transition-Based Dependency Parsing. *Procesamiento del Lenguaje Natural*, 51.

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.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL), pages 149–164.

Jinho D. Choi. 2013. Preparing Korean Data for the Shared Task on Parsing Morphologically Rich Languages. *ArXiv e-prints*, September.

Yoav Goldberg and Michael Elhadad. 2010. Easy first dependency parsing of modern hebrew. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10, pages 103–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 933–939.

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*.

Francesco Mambrini and Marco Carlo Passarotti. 2012. Will a Parser Overtake Achilles? First experiments on parsing the Ancient Greek Dependency Treebank. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT11)*.

Joakim Nivre and Johan Hall. 2010. A quick guide to MaltParser optimization. Technical report, maltparser.org.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

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

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932.

Muntsa Padró, Miguel Ballesteros, Hector Martínez, and Bernd Bohnet. 2013. Finding dependency parsing limits over a large spanish corpus. In *IJCNLP*, Nagoya, Japan. Association for Computational Linguistics.

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.

Mojgan Seraji, Beáta Megyesi, and Joakim Nivre. 2012. Dependency parsers for persian. In *Proceedings of 10th Workshop on Asian Language Resources, at 24th International Conference on Computational Linguistics (COLING 2012)*. ACL Anthology.

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.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *EMNLP*, pages 385–396, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012a. Cross-framework evaluation for statistical parsing. In *EACL*, pages 44–54.

Reut Tsarfaty, Djamé Seddah, Sandra Kuebler, and Joakim Nivre. 2012b. Parsing Morphologically Rich Languages: Introduction to the Special Issue. *Computational Linguistics*, November.

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*.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *COLING*, pages 1391–1400.