ParsPer: A Dependency Parser for Persian

Mojgan Seraji
Uppsala University
Department of Linguistics
and Philology, Sweden
mojgan.seraji@lingfil.uu.se

Bernd Bohnet
Google in London
bohnetbd@gmail.com

Joakim Nivre
Uppsala University
Department of Linguistics
and Philology, Sweden
joakim.nivre@lingfil.uu.se

Abstract

We present a dependency parser for Persian, called *ParsPer*, developed using the graph-based parser in the Mate Tools. The parser is trained on the entire *Uppsala Persian Dependency Treebank* with a specific configuration that was selected by MaltParser as the best performing parsing representation. The treebank's syntactic annotation scheme is based on Stanford Typed Dependencies with extensions for Persian. The results of the ParsPer evaluation revealed a best labeled accuracy over 82% with an unlabeled accuracy close to 87%. The parser is freely available and released as an open source tool for parsing Persian.

1 Introduction

Data-driven dependency parsing is a modern approach that has been successfully applied to develop dependency parsers for different languages (Böhmová et al., 2003; Haverinen et al., 2010; Kromann, 2003; Foth et al., 2014; Vincze et al., 2010). The approach relies solely on a syntactically annotated dataset (*treebank*). However, achieving the best results by this method relies partly on parsing algorithms and selecting the best feature settings. As data-driven dependency parsers induce the syntactic structure backbone in a treebank, they are further, to a great extent, dependent on the representation setup for part-of-speech and dependency labels. These representations are always built upon an already tokenized text. In other words, different tokenizations require different part-of-speech and dependency annotations, which in turn impact the quality of parsing analysis.

Processing and analysis of a language like Persian pose a variety of challenges on various levels, from orthography to syntactic structure (Seraji, 2015). Persian orthography does not follow a consistent standardization. The most challenging cases concern the handling of fixed expressions and various types of clitics. Different variations of writing such cases as attached or detached forms (either delimited with whitespace or zero-width non-joiner)[1] pose challenges for tokenization which in turn impacts the quality of morphological and syntactic analysis. Furthermore, the prevalence of multi-word compound verbs, functioning as a single verb in the form of so called complex predicates or light verb constructions (LVC), is another remarkable feature in the Persian syntactic structure. The situation for automatic analysis of Persian is further complicated by its high degree of free word order.

Therefore, in preparing the treebank data for Persian many difficult decisions had to be made concerning handling fixed expressions and different types of clitics such as pronominal and copula clitics (Seraji, 2015). Fixed expressions in Persian are sometimes written as one single token and sometimes as several tokens. The same happens for different types of clitics. They are sometimes segmented and sometimes unsegmented from the head words. Since the treebank data is taken from the large and open source Uppsala Persian Corpus (UPC),[2] it was impossible to manually separate fixed expressions and clitics from head words in a consistent way in a large corpus like the UPC, containing 2,703,265 tokens. On the other hand, to automatically handle such cases was also impossi-

[1]The zero-width non-joiner (ZWNJ), also known as zero-space or pseudo-space, is a non-printing character used as a boundary inside a word that keeps different affixes and/or clitics unjoined next to the head words.

[2]For a more detailed description of the *Uppsala Persian Corpus* related to the tokenization and morphological annotation see Seraji (2015, Chapter 3). The corpus is open source and freely available at http://stp.lingfil.uu.se/∼mojgan/UPC.html

ble since the process could result in many incorrect conversions by impacting orthographically similar words or endings with different part-of-speech categories.

Hence, to avoid introducing errors in the corpus, fixed expressions are handled as distinct tokens, as long as they were not written as attached forms, and clitics are not separated from the head words but analyzed with special labels at the syntactic level instead. Therefore, in the annotation scheme of the Uppsala Persian Dependency Treebank, apart from 48 dependency labels for basic relations there are 48 complex dependency labels to cover syntactic relations for words containing unsegmented clitics.

Fine-grained annotated data in treebanks normally provides a more complete grammatical analysis which in turn enhances the quality of parsing results. However, complex annotation may not always be beneficial and can impair automatic analysis (Mille et al., 2012; Jelínek, 2014). In this paper, we present different empirical studies where we systematically simplify the annotation schemes for part-of-speech tags and dependency relations within the treebank.

This paper is organized as follows: Section 2 briefly presents the *Uppsala Persian Dependency Treebank*. Section 3 introduces the experimental design. In Section 4, ParsPer is presented and evaluated. Finally, Section 5 concludes the paper.

2 The Uppsala Persian Dependency Treebank

The Uppsala Persian Dependency Treebank (UPDT)[3] (Seraji et al., 2013; Seraji, 2015)[4] is a dependency-based syntactically annotated corpus of contemporary Persian with annotation scheme based on dependency structure. The treebank consists of 6000 annotated and validated sentences, 151,671 tokens, and 15,692 word types with an average sentence length of 25 words. The data is extracted from the open source part-of-speech annotated and validated Uppsala Persian Corpus (UPC) with different genres, containing newspaper articles and texts on various topics such as culture, technology, fiction, and art.

The treebank's syntactic annotation scheme is based on Stanford Typed Dependencies (STD) (de Marneffe and Manning, 2008) with extensions for Persian. This version of STD has a total of 96 dependency relations of which 48 (including 10 new additions to define the syntactic relations in Persian that could not be covered by the primary scheme developed for English) are used for indicating basic relations. The remaining 48 labels are complex, and are used to assign syntactic relations to words containing unsegmented clitics. The treebank is open source and freely available in CoNLL-format.[5]

3 Experimental Design

We carry out two types of experiments, experiments with different parsing representations (we define these as basic experiments henceforth) and experiments with different dependency parsers. For the experiments, the treebank is sequentially split into 10 parts, of which segments 1–8 are used for training (80%), 9 for development (10%), and 10 for test (10%). In the basic experiments, we train MaltParser on the training set and test on the development set. In the latter experiments, we train different parsers on the joint training and development sets (90%) and test on the test set.

We perform the basic experiments under four different conditions. We first experiment with all features and labels that already exist in the treebank. The results achieved by this experiment will be used as the baseline results. We then experiment with different relation sets by removing or merging various feature distinctions in the part-of-speech tagset and the syntactic annotation scheme. The experiments are designed as indicators to see if the conversions help or do not help the parser. In order to get a realistic picture of the parser performance, all these experiments will be performed using automatically generated part-of-speech tags.

All the above experiments will be carried out using MaltParser (Nivre et al., 2006). After discovering the best label set for both part-of-speech tags and dependency relations, we will experiment with other parsers such as MSTParser (McDonald et al., 2005), MateParsers (Bohnet, 2010; Bohnet and Nivre, 2012; Bohnet and Kuhn, 2012), and TurboParser (Martins et al., 2010) to find a state-of-the-art parser for Persian. We evaluate the parsers by experimenting with various feature set-

[3]The treebank is freely available and can be downloaded from http://stp.lingfil.uu.se/∼mojgan/UPDT.html

[4]For the updated version and a more comprehensive description of the *Uppsala Persian Dependency Treebank* guidelines see Seraji (2015, Chapter 5).

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

tings when optional parameter settings for optimization are available and given by the parsers. However, only results for final settings are presented.

The selected state-of-the-art parser for Persian will be called *ParsPer*. For evaluation of ParsPer we first perform a parsing experiment on the treebank data. We then make an independent parsing evaluation by applying the parser on out-of-domain text and present the final results.

3.1 Basic Experiments with MaltParser

To evaluate the overall performance of the parser, we tune parameters to acquire the highest possible results. Thus, we experiment with different algorithms and feature settings to optimize MaltParser. To accomplish the optimization process, we apply MaltOptimizer (Ballesteros and Nivre, 2012). Parser accuracy is evaluated on automatically generated part-of-speech tags.

In order to generate automatic part-of-speech tags, we used the Persian part-of-speech tagger, *TagPer* (Seraji, 2015). However, for the treebank experiments we retrained the tagger to exclude the treebank data to avoid data overlap. The tagging evaluation performed by the new TagPer revealed an overall accuracy of 97.17%, when trained on 90% of the UPC and evaluated on the remaining 10%. The four different experiments include (1) an overall parsing evaluation on full treebank annotation (baseline), (2) an experiment without morphological features in the part-of-speech tagset, (3) an experiment without fine-grained LVC labels, and (4) an experiment without complex labels.

3.1.1 Baseline: Full Treebank Annotation

In this parsing evaluation we trained MaltParser on the UPDT with full part-of-speech tags and all existing dependency relations. The experiment resulted in a labeled attachment score of 78.84% and an unlabeled attachment score of 83.07%. The results will be used as the baseline for subsequent experiments. Labeled recall and precision for the 20 most frequently dependency relations are presented in Table 1.

The results vary greatly across the relation types, with recall ranging from 53.75% for direct object (dobj) to 97.12% for object of a preposition (pobj), and precision varying between 55.37% for clausal complement (ccomp) to 95.57% for object of a preposition (pobj). As indicated in Table 1

DepRel	*Freq. (%)*	*R (%)*	*P (%)*
pobj	16237	**97.12**	**95.57**
poss	16067	89.96	79.28
prep	15643	76.00	74.49
punct	13442	75.04	76.10
amod	9211	90.64	90.72
nsubj	8653	67.60	66.26
conj	8629	67.78	67.78
cc	7657	78.34	77.81
root	5918	81.21	79.87
cop	4427	66.22	73.51
dobj-lvc	4185	91.63	92.06
advmod	4157	70.27	65.82
ccomp	4021	63.54	**55.37**
det	3929	93.79	91.71
dobj	3723	**53.75**	57.01
nn	3339	57.28	79.73
num	2872	92.00	92.00
acc	2535	69.76	69.48
aux	2287	92.14	90.95
complm	2022	77.71	78.61

Table 1: Labeled recall and precision on the development set for the 20 most frequent dependency types in the UPDT, when *MaltParser* is trained on the full treebank annotation. DepRel = Dependency Relations, Freq. = Frequency, R = Recall, P = Precision.

the results for labeled recall and precision for core arguments such as nominal subject (nsubj) and direct object (dobj) are slightly low. This can be explained by the fact that, despite the SOV structure in Persian, subjects and objects may shift order in a sentence. As Persian is a pro-drop language, an object might be placed at the beginning of a sentence (with or without the accusative marker rā) and the subject might either be positioned next or be completely omitted in the sentence but instead be inflected as a personal ending on the verb. There are further cases when subject and object are both omitted but appear as personal endings on the verb, as Persian, syntactically, contains a vast amount of dropped subjects and objects. In all cases, it is hard for the system to identify the correct subject and object in the sentence, which may lead to the dependency relations *nsubj* and *dobj* frequently being interchanged or not being correctly identified. The dependency relation noun compound modifier (nn) is another relation with low recall. We further discovered that the parser had often selected the label possession modifier (poss) instead of *nn*. This can be explained by their usage similarities in the way that both labels are always governed by a noun and used for nouns. The possession modifier (poss) is applied to genitive complements and the compound modifier (nn) to noun compounds (and proper names). However, this difference is not marked in the part-of-speech annotation. Moreover, the number of occurrences of the label *poss* in the training data is higher than the label *nn*, therefore, it is easier for the parser to identify the structure as the dependency relation *poss* than *nn*.

3.1.2 Coarse-Grained Part-of-Speech Tags

The second empirical study was performed in order to select the best part-of-speech encoding set for parsing. In this experiment, we merged all morphological features with their main categories. In this way, feature distinctions that existed for adjective, adverb, noun, and verb were all discarded. For instance, *ADJ_CMPR, ADJ_INO, ADJ_SUP*, and *ADJ_VOC* were merged with *ADJ*, and so forth. Hence, we ran MaltParser on UPDT with 15 auto part-of-speech tags instead of 31. Parsing evaluation revealed the scores of 79.24% for labeled attachment and 83.45% for unlabeled attachment. Comparing the results to the ones obtained by the baseline experiment shows that MaltParser performs better on coarse-grained part-of-speech tags. Table 2 shows the results for labeled recall and precision for the 20 most frequent dependency labels in the UPDT. Again, object of a preposition (pobj) shows the best results with 97.07% for recall and 95.72% for precision, and direct object (dobj) shows the lowest recall and precision, with 52.55% and 55.56%, respectively.

Comparing the recall and precision results of the 20 most frequent dependency labels to the baseline, we see an improvement in many dependency relations. The highest improvement is indicated by the relation clause complement (ccomp) with 3.75% enhancement for recall and 6.3% for precision. The dependency relation clause complement (ccomp), in the treebank, is assigned for complements that are presented by verbs, nouns, or adjectives. Using coarse-grained part-of-speech tags for verbs, nouns, and adjectives leads to higher results. This further assists the relation complementizer (complm) that always introduces a clausal complement (ccomp) achieving 2.29% higher recall and 3.74% higher precision. To follow up the tables, copula (cop) is also one of the dependency relations that shows good improvements specifically for precision, resulting in 1.61% higher recall and 4.82% higher precision. As comparison goes on, results show an improvement for most of the dependency labels. However,

DepRel	*Freq. (%)*	*R (%)*	*P (%)*
pobj	16237	**97.07**	**95.72**
poss	16067	90.18	79.43
prep	15643	76.85	75.57
punct	13442	76.07	76.80
amod	9211	88.69	90.37
nsubj	8653	68.62	64.55
conj	8629	68.85	68.28
cc	7657	78.88	78.14
root	5918	81.38	80.17
cop	4427	67.83	78.33
dobj-lvc	4185	90.23	91.94
advmod	4157	73.31	66.16
ccomp	4021	67.29	61.67
det	3929	94.35	92.78
dobj	3723	**52.55**	**55.56**
nn	3339	57.04	82.46
num	2872	92.92	91.79
acc	2535	69.35	70.20
aux	2287	92.14	89.41
complm	2022	80.00	82.35

Table 2: Labeled recall and precision on the development set for the 20 most frequent dependency types in the UPDT, when *MaltParser* is trained on the UPDT with coarse-grained auto part-of-speech tags. DepRel = Dependency Relations, Freq. = Frequency, R = Recall, P = Precision.

coarse-grained part-of-speech tags have a negative impact on some dependency labels. This is more or less visible in the dependency relations object of a preposition (pobj), adjectival modifier (amod), nominal subject (nsubj), direct object in light verb construction (dobj-lvc), direct object (dobj), noun compound modifier (nn), and auxiliary (aux) which may due to the lack of various distinctions of nouns, adjectives, and verbs. For instance, plural nouns never appear in complex predicates and as seen in the tables direct object in light verb construction (dobj-lvc) has a drop with 1.40% and 0.12% for recall and precision, respectively.

3.1.3 Coarse-Grained LVC Relations

We carried out this experiment by converting all variations of light verb constructions such as *acomp-lvc*, *dobj-lvc*, *nsubj-lvc*, and *prep-lvc* to merely *lvc*. The evaluation showed that the parser achieved a labeled attachment score of 79.46% and an unlabeled attachment score of 83.52%. With respect to the fact that the labeled attachment score is based on the number of correct dependency labels and correct head, the LAS results obtained in this experiment cannot directly be compared to the baseline results, as the two experiments use different label sets. Therefore, output

DepRel	*Freq. (%)*	*R (%)*	*P (%)*
pobj	16237	**97.45**	**95.89**
poss	16067	89.91	79.65
prep	15643	75.04	73.88
punct	13442	76.22	76.72
amod	9211	89.90	90.32
nsubj	8653	70.30	66.92
conj	8629	67.66	67.90
cc	7657	78.88	78.14
root	5918	82.05	81.23
cop	4427	68.10	78.64
lvc	5427	85.92	90.54
advmod	4157	72.64	68.04
ccomp	4021	64.08	57.18
det	3929	94.07	92.76
dobj	3723	**55.26**	**56.79**
nn	3339	58.01	83.28
num	2872	92.92	92.07
acc	2535	70.97	70.97
aux	2287	92.58	92.17
complm	2022	80.57	81.50

Table 3: Labeled recall and precision on the development set for the 20 most frequent dependency types in the UPDT, when *MaltParser* is trained on the treebank with fine-grained auto part-of-speech tags only one light verb construction. DepRel = Dependency Relations, Freq. = Frequency, R = Recall, P = Precision.

differing in this regard can only be evaluated unlabeled. Thus, the unlabeled attachment score that measures the number of tokens with correct head can directly be compared with the baseline. This accordingly means that removing the LVC distinctions from the treebank helps the parser to obtain higher accuracy. As shown in Table 3, the highest recall and precision scores are shown by object of a preposition (pobj), with 97.45% and 95.89% respectively. The lowest recall and precision scores are shown by direct object (dobj) with 55.26% and 56.79%, respectively.

Compared to the baseline results, recall and precision have decreased for the dependency relations prepositional modifier (prep) and adjectival modifier (amod). This can probably be explained by the fact that merging LVC variations makes it harder for the system to select, for instance, a preposition as a prepositional modifier (prep) or an lvc, as well as an adjectival modifier (amod) or an lvc. A striking finding from the results is the outcome achieved by the conversion of different light verb constructions to *lvc*, resulting in 85.92% for recall and 90.54% for precision. Moreover Table 4 shows recall and precision for different types of LVC relations from the baseline experiment when we applied the fine-grained annotated treebank as well as recall and precision of the dependency label *lvc* from Experiment 3 when we tested the treebank with fine-grained part-of-speech tags and merged LVC relations. The entries in the table further present information about frequency of *acomp-lvc*, *dobj-lvc*, *nsubj-lvc*, and *prep-lvc* in Experiment 1[6] as well as the frequency of the label *lvc* in Experiment 3. As presented in Table 4, results for recall and precision are lower than the baseline results for direct object in light verb construction (dobj-lvc) but higher than the results obtained by the adjectival complement in light verb construction (acomp-lvc) and the prepositional modifier in light verb construction (prep-lvc). However, we should be reminded that the label *lvc* covers all types of LVC relations and, as mentioned earlier, it is harder for the system to select a proper label to tokens that sometimes participate in LVC relations and sometimes participate in similar relations to LVC labels such as prepositions that occasionally appear either as the dependency relations prepositional modifier (prep) or as the prepositional modifier in light verb construction (prep-lvc). On the other hand, recall and/or precision for the core arguments nominal subject (nsubj) and direct object (dobj) are improved. In other words, recall is improved by 2.7% and 1.51% for nominal subject (nsubj) and direct object (dobj), respectively. The dependency relation root is further improved by 0.84% for recall and 1.36% for precision. Thus, this merging might be a disadvantage for the relation prepositional modifier (prep) but favors other relations for instance the nominal subject (nsubj). Although providing recall and precision for each and every LVC distinction on a label-by-label basis is most informative, because the label *lvc* covers all types of the LVC variations, we cannot directly compare the results of each with the results obtained by the dependency relation *lvc* in Experiment 3, unless we calculate an overall recall and precision score for all the LVC types in Experiment 1. The results of such statistical calculations revealed an overall recall and precision of 85.55% and 89.16%. Hence, the overall results show that having various types of LVC distinctions in the treebank do not contribute to higher parsing performance.

[6]Given the low frequency of the LVC relations *acomp-lvc*, *nsubj-lvc*, and *prep-lvc* in the treebank, their recall and precision are not presented together with the 20 most frequent dependency types in Table 1.

DepRel	*Freq.*	*R (%)*	*P (%)*
acomp-lvc	681	80.56	78.38
dobj-lvc	4185	91.63	92.06
nsubj-lvc	7	–	–
prep-lvc	554	46.88	78.95
lvc	5427	85.92	90.54

Table 4: Recall and precision for LVC relations with fine-grained predicted part-of-speech tags in Experiments 1 and 3. DepRel = Dependency Relations, Freq. = Frequency, R = Recall, P = Precision.

3.1.4 No Complex Relations

We additionally experimented with modifying all complex syntactic relations that were used for complex unsegmented word forms (words containing unsegmented clitics). In this experiment, all complex dependency relations, containing 48 labels, were merged with basic Persian STD relations, containing 48 labels. The evaluation revealed a labeled attachment score of 79.63% and an unlabeled attachment score of 83.42%. As noted earlier, the results from labeled attachment score do not allow a direct comparison with the ones presented for baseline as the two experiments use different label sets. Hence, the comparison evaluation is considered for the unlabeled attachment score that shows an improvement in parsing performance when simplifying the complex dependency relations. This improvement is understandable as some complex relations[7] such as *ccomp\cpobj*, *ccomp\nsubj*, and so forth, occur only once in the treebank and it is almost impossible for a data-driven machine to learn such rare cases from the given data.

As presented in Table 5, there are variations in recall, ranging from 54.14% for direct object (dobj) to 97.47% for object of a preposition (pobj), and in precision, varying between 56.31% for clausal complement (ccomp) to 96.90% for object of a preposition (pobj). Compared to the baseline, recall and precision for the dependency relations adjectival modifier (amod) and complementizer (complm) have dropped in the figures. The relations *root* and noun compound modifier (nn) as well as punctuation (punct) and auxiliary (aux) further show a decline in recall and precision respectively. This can probably be explained by the way it has been annotated for the complex labels.

[7]Complex relations in the treebank are marked by backslash (\) if they precede the segment carrying the main function and a forward slash (/) if they follow it.

DepRel	*Freq. (%)*	*R (%)*	*P (%)*
pobj	16412	**97.47**	**96.90**
poss	16268	90.27	79.59
prep	15734	76.52	75.62
punct	13442	75.04	75.76
amod	9277	89.75	90.59
nsubj	8847	68.40	66.56
conj	8753	68.63	69.28
cc	7657	79.16	78.41
root	6010	81.17	80.90
cop	4427	66.76	74.55
dobj-lvc	4204	90.76	92.25
advmod	4168	71.62	67.52
ccomp	4105	64.10	**56.31**
det	3929	94.07	93.28
dobj	3862	**54.14**	57.19
nn	3340	56.31	81.98
num	2872	93.23	93.23
acc	2535	71.37	71.08
aux	2287	92.14	90.56
complm	2022	77.14	78.03

Table 5: Labeled recall and precision on the development set for the 20 most frequent dependency types in the UPDT, when *MaltParser* is trained on the treebank with fine-grained auto part-of-speech tags and only basic dependency relations. DepRel = Dependency Relations, Freq. = Frequency, R = Recall, P = Precision.

Removing the information provided by the these relations makes it harder for the parser to achieve high results when assigning these labels. However, the parser shows higher scores for the remaining dependency relations.

3.1.5 Best Parsing Representation

In the recently presented experiments we systematically simplified the annotation schemes for part-of-speech tags and dependency labels. Table 6 presents a summary of the 4 basic experiments we performed. Although the results in the table are presented with labeled and unlabeled attachment score as well as label accuracy score, figures obtained as labeled attachment in Experiments 3 and 4 are not comparable with the one presented in the baseline results, as each performed study uses different dependency relation sets. To conclude the four experiments:

- Using coarse-grained part-of-speech tags in the dependency representation improves parsing performance without losing any information. By using the part-of-speech tagger *TagPer* we can recreate and restore this information at the end once the parsing is done. Thus, fined-grained part-of-speech tags can still be in the output.

Basic Ex.	*LAS (%)*	*UAS (%)*	*LA (%)*
Baseline	78.84	83.07	88.48
CPOS	79.24	83.45	88.43
1 LVC	79.46	**83.52**	88.86
Basic DepRel	**79.63**	83.42	**89.09**

Table 6: Labeled and unlabeled attachment scores, and label accuracy in the model selection resulted from 4 empirical studies when *MaltParser* was trained on UPDT with different simplifications of annotation schemes in predicted part-of-speech tagset and dependency relations. Basic Ex. = Basic Experiments, Baseline = Experiment with a fine-grained annotated treebank, CPOS = Experiment with coarser-grained part-of-speech tags and fine-grained dependency relations, 1LVC = Experiment with fine-grained part-of-speech tags and dependency relations free from distinctive features in light verb construction, and Basic DepRel = Experiment with fine-grained part-of-speech tags and merely basic dependency relations.

- The studies additionally show that simplifying the representation of light verb constructions helps the parser to perform better without loss of important information. In other words, by using coarse LVC, the results become less specific and less informative only with respect to the LVC construction, and show better parsing performance. Furthermore, the *lvc* specification at the end can mostly be recovered from the part-of-speech tags in the output.

- Using merely basic relations might provide a marginal improvement but this is not a sufficient justification to remove them, because by eliminating the complex labels we lose essential information that cannot be recovered by the tagger and this affects the quality of parsing analysis. Applying the treebank with complex relations provides a richer grammatical analysis that boost the quality of parsing results.

These results provided us with a valuable insight about how different morphosyntactic parameters in data influence the parsing analysis. The studies also brought us to the point of how we can select the best configuration for further experiments. In other words, we will use a representation with coarse-grained part-of-speech tags, single LVC representation, and fine-grained dependency relations containing both basic and complex labels (96 labels).

3.2 Experiments with Different Parsers

This part is designed for estimating the performance of different parsers on the best performing data representations selected by MaltParser in the baseline experiments. Hence, we set up the data with the best achieved parameters which are using the automatically generated coarse-grained part-of-speech tags with a single LVC label and the fine-grained dependency relations consisting of 96 basic and complex labels. The treebank is further organized with a different split than in the basic experiments. In other words, we train the parser on the joint training and development sets (90%) and test on the test set (10%). We will experiment with MaltParser (Nivre et al., 2006), MSTParser (McDonald et al., 2005), MateParsers (Bohnet, 2010; Bohnet and Nivre, 2012), and TurboParser (Martins et al., 2010).

For evaluating MaltParser, we used *Nivre's algorithms* as the algorithms were the best parsing algorithms offered by MaltOptimizer during the previous experiments. The parser resulted in scores of 79.40% and 83.47% for labeled and unlabeled attachment, respectively.

For evaluating MSTParser, we used the second-order model with projective parsing as this setting had presented the highest results in the earlier parameter tuning experiments. The parser presented the results of 77.79% for labeled and 83.45% for unlabeled attachment scores.

For experimenting with MateParsers, we trained the graph-based and transition-based parsers on the UPDT with the best parameters selected. The results of Mate experiments showed that the graph-based parser outperformed the transition-based parser, resulting in 82.58% for labeled and 86.69% for unlabeled attachment scores.

For experimenting with TurboParser, we trained the second-order non-projective parser with features for arcs, consecutive siblings and grandparents, using the AD^3 algorithm as a decoder. We adapted the *full* setting as the setting had performed best with our earlier parameter-tuning experiments. The *full* setting enables arc-factored, consecutive sibling, grandparent, arbitrary sibling, head bigram, grand-sibling (third-order), and tri-sibling (third-order) parts. The parser showed the results of 80.57% for labeled and 85.32% for un-

Evaluations	*LAS (%)*	*UAS (%)*	*LA (%)*
MaltParser	79.40	83.47	88.72
MSTParser	77.79	83.45	87.11
MateGraph	**82.58**	**86.69**	**90.55**
MateTrans.	81.72	85.94	89.87
TurboParser	80.57	85.32	88.93

Table 7: Best results given by different parsers when trained on UPDT with auto part-of-speech tags, 1LVC, CompRel in the model assessment. MateGraph. = Mate graph-based, MateTrans. = Mate transition-based

labeled attachment scores.

As shown in Table 7 the graph-based parser in the Mate tools achieves the highest results for Persian. The developed parser will be treated as the state-of-the-art parser for the language and will be called *ParsPer*. The parser will undergo further evaluations which will be presented more in detail in the next section.

4 Dependency Parser for Persian: ParsPer

As results of the previous experiments showed, the graph-based MateParser outperformed MaltParser, MSTParser, and TurboParser obtaining scores of 82.58% and 86.69% for labeled and unlabeled attachment. This means that we need to train the graph-based MateParser, this time, on the entire UPDT with the selected configuration. The developed parser is called *ParsPer*.[8] It is released as a freely available tool for parsing of Persian and is open source under GNU General Public License. The parser will further be evaluated in the next subsection.

4.1 The Evaluation of the ParsPer

To evaluate the performance of the ParsPer we made an independent parsing evaluation by running the parser on out-of-domain text. For this, we used texts from the web-based journal *www.hamshahri.com*. We downloaded multiple texts based on different genres and then randomly picked 100 sentences containing 2778 tokens with an average sentence length of 28 tokens to develop a test set. As our experiment involved some manual work we opted for a small-sized sample to make the evaluation task more feasible. We first created a gold file by manually normalizing the internal word boundaries and character sets and then segmenting the text into sentence and token levels. We then manually annotated the file with part-of-speech and dependency information using the same part-of-speech and dependency scheme that ParsPer was built on to be served as gold.

In this task we performed three different parsing evaluations. First we applied the parser on the automatically normalized, tokenized and tagged text. This is the main experiment in the ParsPer evaluation that also indicates the performance of automatic processing of Persian texts at various levels. Next, we performed two more experiments with the 100 randomly selected sentences in order to analyze the results in a more nuanced way, by experimenting on the sentences when they are manually normalized and tokenized but automatically tagged and then, when they are manually normalized, tokenized, and tagged.

To create our test set for our first experiment, we automatically normalized the 100 sentences using the Persian normalizer *PrePer*,[9] segmented it with *SeTPer*,[10] and tagged with *TagPer*.[11] A comprehensive description of the tools *PrePer*, *SeTPer*, and *TagPer* are given in Seraji (2015, Chapter 4). Then we parsed the automatically tokenized and tagged text with ParsPer. Since the sentences were automatically tokenized, contained 10 tokens fewer than the gold file (the number of tokens in the gold file were 2788).[12] Therefore we could not directly present labeled and unlabeled attachment scores. However, instead, we present labeled recall and precision as well as unlabeled recall and precision. The parsing evaluation revealed a labeled recall and precision of 73.52% and 73.79%, and an unlabeled recall and precision of 81.99% and 82.28%, respectively. As could be expected, the results for labeled recall and precision are low. This is due to the fact that apart from incorrect tokens in the automatically tokenized file there are incorrect part-of-speech tags made by the tagger TagPer that have had a negative impact on the results.

Subsequently, we automatically parsed the manually normalized, tokenized, but automatically tagged text and compared the parsing results with the manually parsed gold text. By this ex-

[8]http://stp.lingfil.uu.se/∼mojgan/parsper-mate.html

[9]http://stp.lingfil.uu.se/∼mojgan/preper.html

[10]http://stp.lingfil.uu.se/∼mojgan/setper.html

[11]http://stp.lingfil.uu.se/∼mojgan/tagper.html

[12]In addition to the 10 fewer tokens, there were two more tokens that were not successfully been normalized by the PrePer in the normalization process and looked differently.

Evaluations	*LR* (%)	*LP* (%)	*UR* (%)	*UP* (%)
AS+AT+AP	73.52	73.79	81.99	82.28
MS+AT+AP	78.50	78.50	86.27	86.27
MS+MT+AP	78.76	78.76	86.12	86.12

Table 8: The evaluation of the *ParsPer* when tested on 100 randomly selected sentences from the web-based journal *Hamshahri*. LR = Labeled Recall, LP = Labeled Precision, UR = Unlabeled Recall, UP = Unlabeled Precision, AS = Automatically Segmented, AT = Automatically Tagged, AP = Automatically Parsed, MS = Manually Segmented, and MT = Manually Tagged.

periment, we wanted to isolate the impact of tagging errors. The evaluation resulted in labeled and unlabeled attachment scores (recall and precision) of 78.50% and 86.27% on the test set with 100 sentences and 2788 tokens. As the results indicate, the unlabeled attachment score is close to the unlabeled attachment score obtained by the parser when evaluated on in-domain text. Furthermore, the unlabeled attachment score is 7.77% higher than the labeled attachment score. This may partly be due to fact that the structural variation for the head nodes is lower than the variation for labels. Moreover, we have a firm structure for the head nodes in the syntactic annotation when invariably choosing content words as head position. This solid structure in turn makes it easier for the parser to learn that after repeatedly seeing it. Hence, the parser assigns the head nodes more accurately than the combinations of head and label. This does not mean that it does not exist a consistent structure for the dependency relations. What we mean is that the number of occurrence of certain cases for dependency relations may not be as many as the number of repeated cases for head structures. This might be perceived as a sparseness by the parser which can directly affect the labeled attachment score. Moreover, the syntactic (non)complexity in the data can have a direct impact on the parser performance.

Finally, we automatically parsed the manually normalized, tokenized, and tagged text and compared the parsing with the manually parsed gold file. The evaluation resulted in a straightforward labeled and unlabeled attachment scores of 78.76% and 86.12% on the test set with 100 sentences and 2788 tokens. The same kind of pattern as in the previous experiment was further found here. In other words, we see nearly similar gap of 7.36% between the labeled and unlabeled attachment scores. Table 8 shows results from different evaluations of the ParsPer.

The comparison of Experiments 1, 2, and 3 shows that tokenization is a greater problem than tagging for syntactic parsing. Whereas a perfectly tokenized text with tagging errors degrades parsing results by less than 1%, errors in tokenization may decrease parsing accuracy as much as 5%. To some extent, this is probably due to additional tagging errors caused by tokenization errors. It is nevertheless clear that tokenization errors disrupt the syntactic structure more than tagging errors do. Adding variations of writing styles (as mentioned earlier) on top of this triggers variations in the tokenization process, which in turn leads to the parser being unable to realize similar sentences with different tokenizations. However, this normally happens when the parser is not familiar with the tokens (or the order of how tokens are represented) in the sentence, which is due to the fact that the structure is not prevalent enough in the training data.

It might be possible to improve the parsing performance by adding to or modifying the part-of-speech tag set as well as eliminating or modifying some structures in the syntactic annotation scheme that are not properly favor the parser. Moreover, one can use joint segmentation and tagging similar to that made for Chinese (Zhang and Clark, 2010). However, this matter will remain for our future research.

5 Conclusion

In this paper, we have presented an open source dependency parser for Persian based on the graph-based parser in the Mate Tools. The dependency parser is called ParsPer and developed on the best performing data representation of the Uppsala Persian Dependency Treebank, selected by MaltParser. The parser resulted in a labeled attachment score of 82.58% and unlabeled attachment score of 86.69%

References

Ballesteros, Miguel and Joakim Nivre (2012). “MaltOptimizer: A System for MaltParser Optimization”. In: *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pp. 833–841.

Böhmová, Alena, Jan Hajič, Eva Hajičocá, and Barbora Hladká (2003). “The Prague dependency treebank”. In: *Treebanks*. Springer Netherlands, pp. 103–127.

Bohnet, Bernd (2010). “Top Accuracy and Fast Dependency Parsing is not a Contradiction”. In: *Coling ’10*, pp. 89–97.

Bohnet, Bernd and Jonas Kuhn (2012). “The Best of Both Worlds: A Graph-based Completion Model for Transition-Based Parsers”. In: *EACL ’12*, pp. 77–87.

Bohnet, Bernd and Joakim Nivre (2012). “A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing”. In: *EMNLP-CoNLL ’12*, pp. 1455–1465.

de Marneffe, Marie-Catherine and Christopher D. Manning (2008). “The Stanford Typed Dependencies Representation”. In: *COLING’08*, pp. 1–8.

Foth, Kilian, Arne Köhn, Niels Beuck, and Wolfgang Menzel (2014). “Because size does matter: The Hamburg dependency treebank”. In: *LREC ’14*, pp. 2326–2333.

Haverinen, Katri, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Filip Ginter, and Tapio Salakoski (2010). “Treebanking Finnish”. In: *TLT ’10*, pp. 79–90.

Jelínek, Tomáš (2014). “Improvements to Dependency Parsing Using Automatic Simplification of Data”. In: *LREC’14*, pp. 73–77.

Kromann, Matthias T. (2003). “The Danish Dependency Treebank and the DTAG Treebank Tool”. In: *TLT ’03*. Brown University Press, pp. 217–220.

Martins, André F. T., Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo (2010). “Turbo Parsers: Dependency Parsing by Approximate Variational Inference”. In: *EMNLP ’10*, pp. 34–44.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič (2005). “Non-Projective Dependency Parsing Using Spanning Tree Algorithms”. In: *HLT-EMNLP ’05*, pp. 523–530.

Mille, Simon, Alicia Burga, Gabriela Ferraro, and Leo Wanner (2012). “How Does the Granularity of an Annotation Scheme Influence Dependency Parsing Performance?” In: *COLING ’12*, pp. 839–852.

Nivre, Joakim, Johan Hall, and Jens Nilsson (2006). “MaltParser: A Data-Driven Parser-Generator for Dependency Parsing”. In: *LREC ’06*, pp. 2216–2219.

Seraji, Mojgan (2015). “Morphosyntactic Corpora and Tools for Persian”. PhD Thesis. Studia Linguistica Upsaliensia 16.

Seraji, Mojgan, Carina Jahani, Beáta Megyesi, and Joakim Nivre (2013). *The Uppsala Persian Dependency Treebank Annotation Guidelines*. Technical Report. Department of Linguistics and Philology, Uppsala.

Vincze, Veronika, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik (2010). “Hungarian Dependency Treebank”. In: *LREC ’10*, pp. 1855–1862.

Zhang, Yue and Stephen Clark (2010). “A fast decoder for joint word segmentation and POS-tagging using a single discriminative model”. In: *EMNLP ’10*, pp. 843–852.