Dependency Parsers for Persian

Mojgan Seraji[1] *Beata Megyesi*[2] *Joakim Nivre*[3]
(1 & 2 & 3) Uppsala University, Department of Linguistics and Philology
`firstname.lastname@lingfil.uu.se`

ABSTRACT
We present two dependency parsers for Persian, MaltParser and MSTParser, trained on the Uppsala PErsian Dependency Treebank. The treebank consists of 1,000 sentences today. Its annotation scheme is based on Stanford Typed Dependencies (STD) extended for Persian with regard to object marking and light verb contructions. The parsers and the treebank are developed simultanously in a bootstrapping scenario. We evaluate the parsers by experimenting with different feature settings. Parser accuracy is also evaluated on automatically generated and gold standard morphological features. Best parser performance is obtained when MaltParser is trained and optimized on 18,000 tokens, achieving 68.68% labeled and 74.81% unlabeled attachment scores, compared to 63.60% and 71.08% for labeled and unlabeled attachment score respectively by optimizing MSTParser.

KEYWORDS: Persian dependency treebank, dependency parsing, Farsi, Persian, MaltParser, MSTParser.

1 Introduction

Data-driven tools for syntactic parsing have been successfully developed and applied to a large number of languages. The existing data-driven syntactic parsers are based on phrase structure, dependency structure, or specific linguistic theories such as HPSG, LFG or CCG. Dependency-based representations have become more widely used in the recent decade, as the approach seems better suited than phrase structure representations for languages with free or flexible word order (Kübler et al., 2009). Dependency parsing, in addition, has been shown to be useful in language technology applications, such as machine translation and information extraction, when detecting the underlying syntactic pattern of a sentence, because of their transparent encoding of predicate-argument structure (Kübler et al., 2009).

This paper presents the adaptation and evaluation of two dependency parsers, MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2005a) for Persian. The parsers are trained on a syntactically annotated corpus developed for Persian; the Uppsala PErsian Dependency Treebank (UPEDT) (Seraji et al., 2012).

The paper is organized as follows. Section 2 briefly describes the structure and the characteristics of the Persian language, followed by a description of the dependency structure and the functional annotation of the Persian dependency treebank on which the data-driven parsers are trained. A short description of MaltParser and MSTParser ends the section. Section 3 introduces the design of our experiments and Section 4 presents the results of the evaluation covering the results of MaltParser and MSTParser, as well as an error analysis for the developed parsers. Finally, Section 5 concludes the paper.

2 Background

2.1 Persian

Persian belongs to the Indo-Iranian languages, a branch of the Indo-European family. The writing system is based on the Arabic alphabet consisting of 28 letters and four additional letters. Persian is written from right to left and Persian morphology is regulated by an affixal system, see more on Persian orthography and morphology in Seraji et al. (2012).

Persian has a SOV word order and is verb final. The head word usually follows its dependents. However, the syntactic relations have a mixed typology, as prepositions always precede their nominal dependent. Sentences consist of an optional subject and object followed by a compulsory verb, i.e., (S) (O) V. Subjects, however, can be placed anywhere in a sentence or they may completely be omitted as Persian is a pro-drop language with an inflectional verb system (where the verb is inflected for person and number). The use and the order of the optional constituents are relatively arbitrary and this scrambling characteristic makes Persian word order highly flexible. Verbs are usually compounds consisting of a preverbal element such as a noun, adjective, preposition, or adverb combined with a bleached or light verb. Light verbs and passive constructions may be split by other intervening elements such as subjunctives, adjectives, future auxiliaries, and negations, that might cause crossing dependencies in syntactic annotation.

2.2 Persian Treebank

Uppsala PErsian Dependency Treebank (UPEDT) (Seraji et al., 2012) is a dependency-based syntactically annotated corpus consisting of 1,000 sentences and 19,232 tokens (14,397 types), which is available in CoNLL-format. The data is taken from the open source, validated corpus UPEC (Seraji et al., 2012) created from on-line material containing newspaper articles and common texts with different genres and topics such as fiction, as well as technical descriptions and texts about culture and art. The corpus is annotated with morpho-syntactic and partly semantic features. The aim is to expand the treebank with 10,000 sentences in the near future by using data-driven dependency parsers for bootstrapping, and manual validation of the annotation.

In the treebank, each head and dependent relation is marked and annotated with functional categories, indicating the grammatical function of the dependent to the head. The treebank annotation scheme is based on Stanford Typed Dependencies (STD) which has become a de facto standard for English today (Marneffe and Manning, 2008). The STD annotation scheme has been applied to Persian and was extended to cover all syntactic relations that are not covered by the original scheme developed for English. Five new labels were added to describe various relations in light verb constructions, and the accusative marker. The added relations are introduced below with a description. The entire annotation scheme can be found in Table 1 where the extended relations introduced for Persian are marked in italic.[1]

***acc*: accusative marker**
An accusative marker is a clitic attached to the direct object of a transitive verb.

***acomp-lvc*: adjectival complement in light verb construction**

[1]An alternative to introducing special relations for the nonverbal elements in light verb constructions would have been to use the *mwe* relation for multi-word expression. However, because light verb constructions are so prevalent in Persian, we chose to distinguish them from other multi-word expressions like compound prepositions and conjunctions.

An adjectival complement in a light verb construction is a preverbal adjective combining with a light verb to form a lexical unit.

***dobj-lvc*: direct object in light verb construction**
A direct object in a light verb construction is a preverbal direct object combining with a light verb to form a lexical unit.[2]

***nsubj-lvc*: nominal subject in light verb construction**
A nominal subject in a light verb construction holds between a preverbal nominal subject combining with a light verb to form a lexical unit (usually with a passive meaning).

***prep-lvc*: prepositional modifier in light verb construction**
A prepositional modifier in a light verb construction is a preverbal prepositional phrase combining with a light verb to form a lexical unit.

In order to increase the size of the treebank, we adapt two freely available dependency parsers that have so far been successfully used for different languages, namely MaltParser (Nivre et al., 2006) and MSTParser (McDonald et al., 2005b), to the Persian dependency treebank.

2.3 Data-Driven Dependency Parsers

MaltParser (Nivre et al., 2006) is an open source data-driven parser generator for dependency parsing. The parser is an implementation of *inductive dependency parsing* (Nivre, 2006) and can be used to develop a parser for a new language given a dependency treebank representing the syntactic relations of that language. The system is characterized as *transition-based*, allowing the user to choose different parsing algorithms and to define optional feature models indicating lexical features, part-of-speech features and dependency type features. The main parsing algorithms available in MaltParser are Nivre's algorithms, including the arc-eager and arc-standard versions described in Nivre (2003) and Nivre (2004), Covington's algorithms, containing the projective and non-projective versions described by Covington (2001), and Stack algorithms, including the projective and non-projective versions of the algorithm described in Nivre (2009) and Nivre et al. (2009). The Covington and the Stack algorithms can handle non-projective trees whereas the Nivre algorithm does not (Ballesteros and Nivre, 2010). For the optimization of MaltParser we used MaltOptimizer (Ballesteros and Nivre, 2010) developed specifically to optimize MaltParser for new data sets with respect to parsing algorithm and feature selection.

MSTParser (McDonald et al., 2005b,a) is also an open source system but based on the graph-based approach to dependency parsing using global learning and exact (or nearly exact) inference algorithms. A graph-based parser extracts the highest scoring spanning tree from a complete graph containing all possible dependency arcs, using a scoring model that decomposes into scores for smaller subgraphs of a tree. MSTParser implements first- and second-order models, where subgraphs are single arcs and pairs of arcs, respectively, and provides different algorithms for projective and non-projective trees.

MaltParser and MSTParser were the top scoring systems in the CoNLL 2006 shared task on multilingual dependency parsing (Buchholz and Marsi, 2006) and has since been applied to a wide range of languages.

[2]Note that this unit may in turn take a direct object. Hence the need to distinguish the light verb object from an ordinary direct object.

Category	**Description**
acc	*accusative marker*
acomp	adjectival complement
acomp-lvc	*adjectival complement in light verb construction*
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
auxpass	passive auxiliary
cc	coordination
ccomp	clausal complement
complm	complementizer
conj	conjunction
cop	copula
dep	dependent
det	determiner
dobj	direct object
dobj-lvc	*direct object in light verb construction*
mark	marker
mwe	multi-word expression
neg	negation modifier
nn	noun compound modifier
npadvmod	noun phrase as adverbial modifier
nsubj	nominal subject
nsubj-lvc	*nominal subject in light verb construction*
nsubjpass	passive nominal subject
num	numerical structure
number	element of compound number
parataxis	parataxis
pobj	object of a preposition
poss	possession modifier
predet	predeterminer
prep	prepositional modifier
prep-lvc	*prepositional modifier in light verb construction*
punct	punctuation
quantmod	quantifier phrase modifier
rcmod	relative clause modifier
rel	relative
root	root
tmod	temporal modifier
xcomp	open clause complement

Table 1: Syntactic relations in UPEDT based on Stanford Typed Dependencies including extensions for Persian.

3 Parsing Persian

We trained the two parsers by applying various algorithms and feature settings on 1,000 sentences of the Persian dependency treebank. 90% of data was used for training and validation and 10% of the data taken from different topics for final test. To find out what impact a PoS tagger has on the parsing results, we trained the parsers by using gold standard PoS tags taken from UPEC, as well as automatically generated morphological features during training and test. For the automatic morphological annotation, we used TagPer, a freely available part of speech tagger for Persian (Seraji et al., 2012). TagPer was developed by using HunPoS (Halácsy et al., 2007) trained on UPEC consisting of 2,698,274 tokens and has proven to give state-of-the-art accuracy of 97.8% for PoS tagging of Persian (Seraji et al., 2012). TagPer was retrained for our experiments in order to exclude treebank data to avoid data overlap. The results performed by the new TagPer revealed an overall accuracy of 96.1%.[3]

3.1 MaltParser

In order to obtain the highest possible accuracy, given the small data set, we experimented with different algorithms and feature settings to optimize MaltParser. For the optimization, we used the freely available optimization tool for MaltParser, namely MaltOptimizer (Ballesteros and Nivre, 2010). We also used the parser out of the box with its default settings. In addition, we trained the parser with gold standard PoS tags as well as with auto generated PoS tags. In all experiments we used 90% of the treebank data set for training and validation by applying 10-fold cross-validation.

The results are shown in Table 2. Parser optimization leads to improved labeled and unlabeled attachment scores and label accuracy. MaltOptimizer ended up with different algorithms for the different folds during cross validation with best results shifting between *Nivre's algorithms* and *Covington's algorithms*.

We can note that using gold standard PoS features during training and test (DGG and OGG) gives higher attachment scores compared to auto generated PoS tags (DAA and OAA) independently of whether the parser is optimized or not. Higher results are obtained when the parser is trained and tested on automatically genererated PoS features (DAA and OAA), a scenario realistic for parsing new running texts, while accuracy is lowest when the parser is trained on gold-standard features but parses texts that are automatically annotated by a PoS tagger (DGA and OGA).

3.2 MSTParser

To apply MSTParser to Persian, we used the same experiment settings and the same data division, to keep the same criteria as we used for MaltParser. In other words, we continued the validation phase by training MSTParser with its default settings (projective, order 1) and optimized the parser with regards to feature order (order 1 and 2) and non-projective structures. Thus, we combined the training strategies by running the parser in four different experimental settings: projective-order1 (default), projective-order2, non-projective-order1, and non-projective-order2.

The cross-validation results for MSTParser as reported in Table 3 reveal the scores between 62.35% to 65.99% for labeled attachment, and 69.52% to 72.68% for unlabeled attachment.

[3]The accuracy of the new TagPer is lower due to the treatment of lvc-construction, multiword expressions, as well as the further correction of UPEC. These corrections were required for our treebank development.

	Default			Optimized		
Including Punct.	**DGG**	**DGA**	**DAA**	**OGG**	**OGA**	**OAA**
Labeled Attachment	68.46	64.80	66.17	**70.39**	66.14	69.37
Unlabeled Attachment	74.07	70.36	71.78	**75.49**	72.19	74.12
Label Accuracy	80.19	78.52	79.81	**83.19**	79.72	81.50

Table 2: Labeled and unlabeled attachment score, and label accuracy score including punctuation of MaltParser in the model selection with different feature settings. DGG = Default with Gold PoS tags in the training and the test set, DGA = Default with Gold PoS tags in the training and Auto PoS tags in the test set, DAA = Default with Auto PoS tags in the training and the test sets, OGG = Optimized with Gold PoS tags in the training and the test set, OGA = Optimized with Gold PoS tags in the training and Auto PoS tags in the test set, and OAA = Optimized with Auto PoS tags in the training and the test set

The optimized versions (OGG, OGA, and OAA), as could be predicted, achieved higher accuracy compared to the default settings (DGG, DGA, and DAA). Highest results on average are achieved by using only projective structures and feature order 2. However, the optimization scores varied slightly between the structures projective-order 2, non-projective-order1 and non-projective-order2 across the folds.

Like MaltParser, the gold standard PoS features (DGG and OGG) give higher attachment scores compared to the auto generated PoS tags (DAA and OAA) independently of whether the MSTParser is optimized or not. Lowest parser performance is obtained when the parser is trained on gold PoS features but tested on automatically generated ones.

	Default			Optimized		
Including Punct.	**DGG**	**DGA**	**DAA**	**OGG**	**OGA**	**OAA**
Labeled Attachment	65.58	62.35	63.78	**65.99**	62.59	64.44
Unlabeled Attachment	72.19	69.52	71.06	**72.68**	69.99	71.80
Label Accuracy	80.37	77.54	78.57	**80.47**	77.43	79.03

Table 3: Labeled and unlabeled attachment score, and label accuracy including punctuation of MSTParser in the model selection with different feature settings (for explanation of the features see Table2).

4 Results

For the final test, we trained the parsers on 90% of the data and tested on a separated, previously unseen test set of 10%. We run the parsers using the feature settings based on their best performance during the validation phase. For MaltParser, we used *nivreeager* algorithm and for the MSTParser, we applied projective training and feature order 2, as these settings had shown the best results on average during the validation phase. The results are given separately with punctuations and without, as shown in Table 4 and Table 5.

Similar to the development experiments, MaltParser obtains highest accuracy in all experiments. The labeled and unlabeled attachment scores reach best result with gold standard PoS tags. However, using automatically generated PoS features during training and test gives higher accuracy compared to when training the parsers on gold-standard PoS but using automatically generated PoS tags on the test data.

	MALT			MST		
Including Punct.	**OGG**	**OGA**	**OAA**	**OGG**	**OGA**	**OAA**
Labeled Attachment	**68.68**	64.05	65.77	63.60	59.94	60.76
Unlabeled Attachment	**74.81**	70.93	73.39	71.08	68.09	68.91
Label Accuracy	**80.64**	76.91	77.50	77.73	74.96	75.78

Table 4: Labeled and unlabeled attachment score, and label accuracy score including punctuation in the model assessment with different feature settings.

	MALT			MST		
Ignoring Punct.	**OGG**	**OGA**	**OAA**	**OGG**	**OGA**	**OAA**
Labeled Attachment	**68.57**	63.54	65.33	63.54	59.20	60.31
Unlabeled Attachment	**75.55**	71.38	73.94	72.06	68.48	69.59
Label Accuracy	**78.28**	73.94	74.70	74.62	71.47	72.40

Table 5: Labeled and unlabeled attachment score, and label accuracy score ignoring punctuation in the model assessment with different feature settings.

In order to provide a more fine-grained analysis of the parsing results for the individual structures, Table 6 displays labeled recall and precision for the 15 most frequently occurring dependency types. As we see, the results vary greatly for both parsers across the relation types. For MaltParser, there is a variation for recall ranging from 46.94% for clausal complement (ccomp) to 91.43% for determiner (det), and for precision from 51.11% for clausal complement to 88.89% for adjectival modifier (amod). For MSTParser, recall is ranging from 34.69% for clausal complement (ccomp) to 92.31% for object for a preposition (pobj), and precision is varying between 36.17% for clausal complement (ccomp) to 88.89% for object for a preposition (pobj).

MaltParser and MSTParser show similar results only for a few relations, which include possessive (poss) and adjectival (amod) modifier, copula (cop), direct object in light verb construction (dobj-lvc), and determiner (det). MaltParser assigns the relation determiner (det) followed by adjectival modifier (amod) to be the candidates of having the highest scores for both recall and precision, while MSTParser assigns object of a preposition (pobj) with the highest F-score. Furthermore, conjunction (conj), direct object (dobj), and the clausal complement (ccomp), are selected as the most erronous relations, although there are differences in the precision and recall figures achieved by the parsers. MaltParser outperforms MSTParser in all cases except for the detection and correctness of objects of a preposition (pobj), and slightly better recall for prepositional modifiers (prep) and coordinating conjunctions (cc).

5 Conclusion

In this paper we have presented two parsers developed for Persian using existing data-driven dependency parsers, MaltParser and MSTParser trained on the Uppsala PErsian Dependency Treebank. The development of the parsers and the treebank has been accomplished simultanously using bootstrapping. As the next step and the highest priority of our future directions involve further annotation and development of our treebank to improve the parsing accuracy.

		MALT		MST	
DepRel	**Freq**	**Rec**	**Prec**	**Rec**	**Prec**
pobj	104	89.42	86.11	92.31	88.89
prep	103	62.14	62.14	63.11	59.09
root	101	80.20	72.32	68.32	69.00
nsubj	99	61.62	55.45	51.52	50.00
poss	97	77.32	65.79	77.32	59.06
advmod	65	63.08	69.49	53.85	60.34
conj	58	50.00	54.72	36.21	44.68
dobj	56	51.79	61.70	41.07	50.00
cc	55	61.82	68.00	63.64	62.50
amod	54	88.89	88.89	83.33	78.95
cop	53	64.15	72.34	64.15	57.63
ccomp	49	46.94	51.11	34.69	36.17
dobj-lvc	43	88.37	69.09	88.37	74.51
det	35	91.43	84.21	91.43	84.21
acc	34	82.35	82.35	73.53	73.53

Table 6: Labeled recall and precision achieved by MaltParser and MSTParser for the 15 most frequent dependency types in the test set.

Acknowledgments

We would like to thank Jon Dehdari for his generosity, sharing his annotation scheme and a set of annotated sentences as a commencement for our treebank annotation effort. We are also grateful to Carina Jahani for her fruitful comments and deliberation regarding Persian grammar. Furthermore, we are particularly grateful to Recorded Future Inc. for their financial support in developing the treebank. The first author has been partly supported by Swedish Graduate School in Language Technology (GSLT).

References

Ballesteros, M. and Nivre, J. (2010). Maltoptimizer: A system for maltparser optimization. In *Proceedings of the 8th international conference on Language Resources and Evaluation (LREC)*, pages 833–841.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.

Covington, M. A. (2001). A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (ACL)*, pages 209–212.

Kübler, S., McDonald, R., and Nivre, J. (2009). *Dependency Parsing*. Morgan & Claypool Publishers series.

Marneffe, M.-C. D. and Manning, C. D. (2008). Stanford typed dependencies representation. In *Proceedings of the COLING'08 Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

McDonald, R., Crammaer, K., and Pereira, F. (2005a). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language (HLT/EMNLP)*, pages 523–530.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Nivre, J. (2004). Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pages 50–57.

Nivre, J. (2006). *Inductive Dependency Parsing*. Springer.

Nivre, J. (2009). Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.

Nivre, J., Hall, J., and Nilsson, J. (2006). 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.

Nivre, J., Kuhlmann, M., and Hall, J. (2009). An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 73–76.

Seraji, M., Megyesi, B., and Nivre, J. (2012). A basic language resource kit for persian. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey.