An Empirical Study on the Effect of Morphological and Lexical Features in Persian Dependency Parsing

Mojtaba Khallash, **Ali Hadian** and **Behrouz Minaei-Bidgoli**
Department of Computer Engineering
Iran University of Science and Technology
{khallash,hadian}@comp.iust.ac.ir, b_minaei@iust.ac.ir

Abstract

This paper investigates the impact of different morphological and lexical information on data-driven dependency parsing of Persian, a morphologically rich language. We explore two state-of-the-art parsers, namely MSTParser and MaltParser, on the recently released Persian dependency treebank and establish some baselines for dependency parsing performance. Three sets of issues are addressed in our experiments: effects of using gold and automatically derived features, finding the best features for the parser, and a suitable way to alleviate the data sparsity problem. The final accuracy is 87.91% and 88.37% labeled attachment scores for MaltParser and MSTParser, respectively.

1 Introduction

Researchers have paid a lot of attention to data-driven dependency parsing in recent years (Bohnet and Kuhn, 2012; Bohnet and Nivre, 2012; Ballesteros and Nivre, 2013). This approach is language-independent and is solely dependent on the availability of annotated corpora. Using data-driven parsers for some languages requires careful selection of features and tuning of the parameters to reach maximum performance. Difficulty of dependency parsing in each language depends on having either free word order or morphological information. Languages with free word order have a high degree of freedom in arranging the words of a sentence. Consequently, they usually have a high percentage of non-projective structures. Morphology is determined by large inventory of word forms (Tsarfaty et al., 2010).

According to the results from CoNLL shared task 2007, languages are classified to three classes, namely *low*, *medium* and *high* accuracy languages. Among them, low-accuracy languages have high degree of free word order along with inflection (Nivre et al., 2007a). Languages which are more challenging in parsing are called morphologically rich languages (MRLs). In MRLs, multiple levels of information, concerning syntactic units and relations, are expressed at the word-level (Tsarfaty et al., 2010).

Free word order can be handled by non-projective parsing algorithms via either post-processing the output of a strictly projective parser (Nivre and Nilsson, 2005), combining adjacent (Nivre, 2009) or non-adjacent sub-structures (McDonald et al., 2005). Nevertheless, there is no general solution for resolving rich morphology issue and hence many researcher focus on features of a specific language. Most data-driven dependency parsers do not use any information that is specific to the language being parsed, but it is shown that using language specific features has a crucial role in improving the overall parsing accuracy (Ambati et al., 2010a).

Persian is an Indo-European language that is written in Perso-Arabic script (written from right to left). The canonical word order of Persian is SOV (subject-object-verb), but there are a lot of frequent exceptions in word order that turn this language into a free word order language (Shamsfard, 2011). This language has a high degree of free word order and complex inflections. As an example of rich morphology, there are more than 100 conjugates and 2800 declensions for some lemmas in Persian (Rasooli et al., 2011).

Dependency treebank for Persian (Rasooli et al., 2013) language has newly become available. Due to the lack of deep research on dependency parsing in Persian, we establish some baselines for dependency parsing performance. We also conduct a set of experiments in order to estimate the effect of errors in morphological disambiguation on the parsers. We show that with two simple changes to the input data, performance of the two parsers can be improved for both gold (manually annotated) and predicted data.

The remainder of the paper is organized as follows. Section 2 presents a brief overview of recent studies on parsing morphologically rich languages. In section 3, we introduce available morphological features annotated in our experiments. Section 4 describes the experimental setup, including corpus and parsers we use, and presents our experiments. Experimental evaluation and analysis of parsing errors are demonstrated in Section 5. Finally, we draw conclusions and suggest future work in Section 6.

2 Related work

Many studies have been done on using morphological features for parsing morphologically rich languages, (e.g. Bengoetxea and Gojenola (2010), Seeker and Kuhn (2013), etc.). Koo et al. (2008) introduce cluster-based features that incorporate word clusters derived from a large corpus of plain text, to improve statistical dependency parsing for English and Czech. Agirre et al. (2011) use lexical semantic information derived from WordNet.

Marton et al. (2011) augment the baseline model for Arabic with nine morphological features. They show that using predicted features causes a substantial drop in accuracy while it greatly improves performance in the gold settings. They show that using noisy morphological information is worse than using nothing at all. Same phenomenon is reported for Hebrew (Goldberg and Elhadad, 2010), except that using morphological-agreement feature improves the accuracy of both gold and predicted morphological information.

Another interesting research direction is to find the most beneficial features for dependency parsing for each language. Ambati et al. (2010b) explored the pool of features for Hindi through a series of experiments. In their setting, features are incrementally selected to create the best parser feature set. In Korean, Choi and Palmer (2011b) focus on feature extraction and suggest a rule-based way of selecting important morphemes to use only these as features to build dependency parsing models.

For the Persian language, Seraji et al. (2012b) investigated state-of-the-art dependency parsing algorithms on UPDT[1] (Seraji et al., 2012a). They test three feature settings, namely gold POS tags for both the training and the test sets (GG), gold POS tags for the training set and auto-generated POS tags for the test set (GA), and auto-generated POS tags for both the training and the test sets (AA). The best result is obtained in GG setting with 68.68% and 63.60% LAS, for MaltParser (Nivre et al., 2007b) and MSTParser (McDonald et al., 2005) respectively. Using AA and GA settings show worse results than GG, namely 2.29% and 3.66% drop in accuracy for MaltParser, and 1.8% and 3.23% drop for MSTParser. They only explore the effect of gold and non-gold POS tags with a small treebank with about 1,300 sentences. We apply GG and AA settings in our experiments on a larger treebank that contains richer morphological information. We define pool of 10 morphological and lexical semantic features in order to create the best feature set for the parser.

3 Features of Persian

In this section, among possible morphological and semantic features that exist in Persian, we briefly review a subset of them that is either annotated in Persian dependency treebank (Rasooli et al., 2013) or is available from other studies.

3.1 Features from Treebank

Table 1 represents the features available in the Persian dependency treebank, along with possible values for each feature.

[1]Uppsala Persian Dependency Treebank

Feature	Values
Attachment	{NXT, PRV, ISO}
Animacy	{animate, inanimate}
Number	{singular, plural}
Person	{1, 2, 3}
Comparison	{positive, comparative, superlative}
TMA	see Table 2

Table 1: Description of features in Treebank

In some special cases, we have to break a word into smaller parts in order to capture the syntactic relations between the elements of the sentence. For example, the two-word sentence صدایم کرد 'sedAyam kard' (called me), consist of three morphemes: صدا (calling), یم (me), and کرد (to do) that have NXT (attached to the next word), PRV (attached to the previous word), and ISO (isolated word) attachment, respectively.

Person and number play a role in constraining syntactic structure. Verbs usually agree with subject in person and number (Shamsfard, 2011). This agreement is useful feature to detect subject of sentence. for example in "پسر، بچه‌ها رفتند" (hey boy, the kids are gone) sentence, both *boy* and *kids* are noun, but only kids has number agreement with verb.

Tense, mood, and aspect are not separately annotated in the treebank, but they can be induced from the TMA value. Table 2 shows the conversion table which consists of 14 valid TMA values. There is not a unique mapping from TMA to aspect, because in some conditions there is interference between the aspects. For example, in indicative imperfective perfect, the verb has perfect or continuous aspects.

3.2 Automatic Semantic Features

Word Clusters [WC] We use all the words of the treebank as inputs to the modified version of Brown clustering algorithm (Liang, 2005). In order to tune the parameters for the two parsers, we tweak the cluster count from 50 to 300 with steps of 50, and bit strings from 4 to 14. Finally, we choose 300 clusters and 6–bit strings for MaltParser and 150 clusters and 10–bit strings for MSTParser[2].

[2]https://github.com/mojtaba-khallash/word-clustering

TMA	Meaning	Mood	Tense
HA	Imperative	Imp.	Pres.
AY	Indicative Future	Ind.	Fut.
GNES	Indicative Imperfective Perfect	Ind.	Past
GBES	Indicative Imperfective Pluperfect	Ind.	Past
GES	Indicative Imperfective Preterit	Ind.	Past
GN	Indicative Perfect	Ind.	Past
GB	Indicative Pluperfect	Ind.	Past
H	Indicative Present	Ind.	Pres.
GS	Indicative Preterit	Ind.	Past
GBESE	Subjunctive Imperfective Pluperfect	Sub.	Past
GESEL	Subjunctive Imperfective Preterit	Sub.	Past
GBEL	Subjunctive Pluperfect	Sub.	Past
HEL	Subjunctive Present	Sub.	Pres.
GEL	Subjunctive Preterit	Sub.	Past

Table 2: Tense/Mood/Aspect types in Persian verbs. Imp., Ind., Sub., Fut., and Pres. stand for imperative, indicative, subjunctive, future and present, respectively.

Semantic Verb Clustering [VC]: Semantic verb cluster is a generalization over verbs according to their semantic properties that capture large amounts of verb meaning without defining details for each verb. Aminian et al. (2013) clustered 1082 Persian verbs into 43 (fine-grained) semantic classes using spectral clustering. For each verb in the treebank, we included the corresponding cluster ID if the verb exists in the list of clustered verbs[3].

Synset Identifier [SID]: FarsNet (Shamsfard et al., 2010) is a lexical ontology for the Persian language that contains approximately 10000 synsets. For each word in the treebank, we look up for possible synsets in FarsNet. If any synset is found, we add the ID of the first synset to our feature set. About 59% of words in the treebank were supplied with a synset.

Semantic File [SF]: In English WordNet, each synset belongs to a unique semantic file. There is a total of 45 semantic files (1 for adverbs, 3 for adjectives, 15 for verbs, and 26 for nouns), based on syntactic and semantic categories (Agirre et al., 2011). FarsNet has a mapping to those of WordNet synsets. We use both synsetID and semantic files as instances of fine-grained and coarse-grained semantic representations, respectively. Thus, we can

[3]https://github.com/mojtaba-khallash/verb-spectral-cluster

learn what level of granularity in semantic features can help improve performance of the parser[4].

4 Experiments

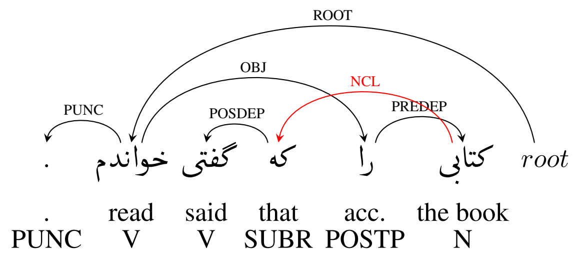
Corpus Persian dependency treebank version 1.0 (Rasooli et al., 2013) is a freely-available resource[5] with about 30,000 sentences, and half a million tokens, annotated with syntactic roles in addition to morpho-syntactic features. The annotation employs 17 coarse-grained and 30 fine-grained POS tags, 22 morphological feature values and 43 dependency labels. 21.93% of the sentences and 2.47% of the edges are non-projective.

Table 3 provides statistical properties of Persian dependency treebank, compared to UPDT[6]. In Persian dependency treebank, syntactic and/or morphological features are represented as key-value pairs separated by vertical bars ('|'), while in UPDT, they are represented as a single atomic feature.

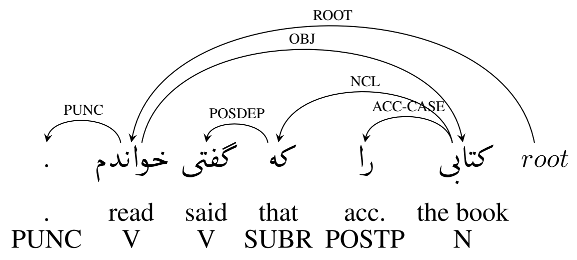
Treebank	**Persian DT**	**UPDT**
Tok	498081	151671
Sen	29982	6000
AvgSL	16.61	25.28
Lem	yes	no
CPoS	17	15
PoS	30	30
MSF	22	30
Dep	43	48
NPT	2.47%	0.17%
NPS	21.93%	2.73%

Table 3: Comparison of UPDT (Seraji et al., 2012a) and Persian dependency treebank (Rasooli et al., 2013). Tok = number of tokens; Sen = number of sentences; AvgSL = Average sentence length; Lem = lemmatization present; CPoS = number of coarse-grained part-of-speech tags; PoS = number of (fine-grained) part-of-speech tags; MSF = number of morphosyntactic features (split into atoms); Dep = number of dependency types; NPT = proportion of non-projective dependencies/tokens (%); NPS = proportion of non-projective dependency graphs/sentences (%)

The data is split into standard train, development and test sets by the ratio of 80-10-10 percent in the CoNLL dependency format. Furthermore, the treebank is released in two representations with little changes in their annotations. A sample comparison between the two annotations is shown in Figure 1. In the first representation, which is manually annotated, the accusative case marker را /rɑ/ is supposed to be the head of the object plus rɑ. In the second representation, which is an automatic conversion of the first one obtained by reverse ordering the manual annotation, rɑ is not the head of the object word. Instead, rɑ is regarded as the accusative case marker for the direct object.


ROOT
OBJ
NCL
PUNC
POSDEP
PREDEP
. خواندم گفتی که را کتابی root
. read said that acc. the book
PUNC V V SUBR POSTP N

(a) First representation: Manually annotating accusative case marker را as object of the sentence


ROOT
OBJ
NCL
PUNC
POSDEP
ACC-CASE
. خواندم گفتی که را کتابی root
. read said that acc. the book
PUNC V V SUBR POSTP N

(b) Second representation: Automatic conversion of first representation. The accusative case marker را depends on original object of the sentence.

Figure 1: Two representation of object-verb relation for "I read the book that you mentioned." (Rasooli et al., 2013).

Evaluation metric The most commonly used metrics for dependency parsing are unlabeled attachment score (UAS), labeled attachment score (LAS) and label accuracy (LA). UAS is the proportion of words that are assigned the correct head, LAS is the proportion of words that are assigned the correct head and dependency type, and LA is the proportion of words that are assigned the correct dependency

[4]`https://github.com/mojtaba-khallash/semantic-tagger`

[5]`http://www.dadegan.ir/en`

[6]Freely available at `http://stp.lingfil.uu.se/~mojgan/UPDT.html`

type. We use LAS as our evaluation metric and take punctuation into account as for evaluating out parsing results. We use McNemars statistical significance test as implemented by (Nilsson and Nivre, 2008), and denote $p < 0.05$ and $p < 0.01$ with $^{+}$ and $^{++}$, respectively.

Parsers We use two off-the-shelf data-driven parsers, namely MaltParser (Nivre et al., 2007b) and MSTParser (McDonald et al., 2005), which are the two state-of-the-art dependency parsers that represent dominant approaches in data-driven dependency parsing.

MaltParser[7] is based on a transition-based approach to dependency parsing. Transition-based approach is based on transition systems for deriving dependency trees, that greedily searches for highest scoring transitions and uses features extracted from parse history to predict the next transition (Choi and Palmer, 2011a). We use MaltParser 1.7.1 along with nine different parsing algorithms. In order to select the best algorithm and tune the parameters of MaltParser, we use *MaltOptimizer* (Ballesteros and Nivre, 2012) on the whole of training data. MaltOptimizer analyzes data in three-phase optimization process: data analysis, parsing algorithm selection, and feature selection.

MSTParser[8] is based on a graph-based approach to dependency parsing. The algorithm searches globally in a complete graph to extract a spanning tree during derivations using dynamic programming. We use MSTParser 0.5 which has two implementations of maximum spanning tree (MST) algorithm with projective and non-projective models[9].

Baseline Experiments We run three phases of MaltOptimizer on the training set in order to find the best parsing algorithm in MaltParser. The first phase validates the data and gains 84.02% LAS with the default settings. In the second phase, using non-projective version of the Covington algorithm, which has the best accuracy, and after parameter tuning, 85.86% LAS was obtained. In the third phase, the feature model was optimized and by tuning the regularization parameter of the multiclass SVM; it led to 87.43% LAS. Finally, we trained the best algorithm with optimized settings on training set and parsed on development set, thereby we reached 87.70% LAS as the baseline of MaltParser.

We tested four parsing algorithms that exist in MSTParser and as a result, non-projective algorithm with a second-order feature decoder gave 88.04% LAS, which shows the highest improvement. Therefore, we selected that as our baseline for MSTParser.

The baselines are obtained on the first representation of the treebank. We found baselines for the second representation of the treebank on the development set. Results are compared in Table 4.

The first representation performs better than the second one. This was expected before, since rɑ is a constant word that is annotated as the object of a sentence in the first representation. This helps parsers to find the object in a sentence. Moreover, as shown in Figure 1, rɑ is closer to the verb than the direct object, hence it has more chance to select.

Representation	**Malt**	**MST**
First	87.70	88.04
Second	87.22 (-0.48)	87.03 (-1.01)

Table 4: Comparison of two representations of Persian treebank

Results In our experiments, we use the first representation of treebank with algorithms and new configurations presented in previous paragraph. For all experiments in this section, we use training and development sets of the treebank. In order to study the effects of morphology in dependency parsing of Persian, we organize experiments into three types of challenges which are presented by Tsarfaty et al. (2010): architecture and setup, representation and modeling, and estimation and smoothing.

Architecture and Setup When using dependency parsing on real-world tasks, we usually face with sentences that must be tokenized, lemmatized, and tagged with part of speech and morphological information to offer those information as input features to the parsing algorithms. Bijankhan corpus (Bijankhan, 2004) is the first manually tagged Persian

[7]`http://www.maltparser.org/`

[8]`http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`

[9]We developed an all-in-one *dependency parsing toolbox* that integrates different dependency parsing algorithms: `https://github.com/mojtaba-khallash/dependency-parsing-toolbox`

corpus that consists of morpho-syntactic and minimal semantic annotation of words. It is commonly used to train POS tagger, but its POS tagset is different from tagset of the treebank that we use. Sarabi et al. (2013) introduce PLP Toolkit which is a comprehensive Persian Language Processing (PLP) toolkit that contains fundamental NLP tools such as tokenizer, POS tagger, lemmatizer and dependency parser. They merged the POS tagset of 10 million words from bijankhan corpus with Persian dependency treebank in order to create a bigger corpus with the same tagset. They choose the tagset of Persian dependency treebank as the base setting and convert Bijankhan tagset to them. They have 11 coarse-grained and 45 fine-grained POS tags. PLP POS tagger can automatically recognize three morphological features, namely number, person, and TMA. TMA values of the PLP tool are not the same as Persian dependency treebank. Despite 14 possible TMA values in dependency treebank (Table 2), only four out of the 14 values exist in PLP (AY, GS, H, and HA), because there is no other value in Bijankhan tagset for verbs. The accuracy of PLP POS tagger on the fine grained tagset is about 98.5%. We use this tagger and apply it on our training, development, and test data. Results from these experiments are presented in Table 5.

POS tags type	Malt	MST
Gold	87.70	88.04
Predicted	86.98 (-0.72)	86.81 (-1.23)

Table 5: Effect of gold vs. predicted POS tags and morphological information in dependency parsers for Persian.

Representation and Modeling In our experiment, we use ten features of morphological and semantic information. Using a forward selection procedure, the best feature set for each parser can be found. Beside morphological features which exist in the treebank (**Attachment [A]**, **Person [P]**, **Number [N]**, **TMA**), we add **Tense [T]** and **Mood [M]** with a simple conversion table, shown in Table 2, based on the value of TMA.

Table 6 shows the effect of each feature for MaltParser and MSTParser parser. For the former, mood with slight differences achieves the best result and for the latter, TMA has the highest accuracy than other features. TMA and two derivate features, namely T and M, stands at the top of this ranking, and four semantic features are placed in the middle. This means that our newly added features can help to improve performance of each parser.

Feature	Malt	Feature	MST
Baseline	87.70	Baseline	88.04
M	**87.77**	**TMA**	**88.21**$^{+}$
TMA	87.77	M	88.17
T	87.73	P	88.09
SF	87.70	T	88.04
WC	87.69	N	88.04
VC	87.68	SID	88.03
SID	87.67	SF	88.03
A	87.67	WC	88.02
P	87.66	VC	87.98
N	87.65	A	87.93

Table 6: Effect of each feature on two parsers

In the next steps, we incrementally add one feature to the best result from previous step. As shown in Table 7, combination of M and SF obtains the best result for MaltParser (87.81%), while for MSTParser, combination of TMA and WC is the best (88.25%). In the second step, adding one semantic feature gets the best result. By trying to continue this approach, we do not see any improvement in the accuracy for both parser[10].

Feature	Malt	Feature	MST
{M,SF}	**87.81**	**{TMA,WC}**	**88.25**
{M,T}	87.79	{TMA,SID}	88.21
{M,VC}	87.78	{TMA,N}	88.16
{M,TMA}	87.77	{TMA,P}	88.14
{M,N}	87.76	{TMA,M}	88.13
{M,WC}	87.75	{TMA,A}	88.11
{M,A}	87.75	{TMA,T}	88.11
{M,P}	87.73	{TMA,VC}	88.07
{M,SID}	87.69	{TMA,SF}	88.05

Table 7: Combinations of two features

[10]`https://github.com/mojtaba-khallash/treebank-transform`

Estimation and Smoothing Using a few training data, especially for languages with rich morphology, lexical features may infrequently appear during training. In MRLs like Persian, due to many feature combination by the inflectional system, we face a high rate of out-of-vocabulary. There are some ways to cope with this problem:

- **Replacing word forms by lemma:** Lemma of a word has less data sparsity than word form.
- **Number Normalization** This is the default approach in MSTParser, in which each number is replaced by a constant. We apply this approach for numbers written either in English or Persian scripts.
- **Word Clustering** and **Semantic File**: The cluster ID of a word or its semantic file can be used instead of the original word form. These are two ways to categorize words into a group bigger than their lemma.

Table 8 illustrates the effect of each smoothing method on the accuracy for parsing MaltParser and MSTParser. For MaltParser, number normalization is the only technique that improves the accuracy. For MSTParser, replacing word forms by lemma and number normalization improves the accuracy. In the case of MSTParser, we apply each method separately and simultaneously on the development set, but replacing word forms by lemma gets the best improvement, and hence we use it in our final configuration.

Smoothing	**Malt**	**MST**
Baseline	87.70	88.04
Replacing word forms by lemma	87.38	**88.10**
Number Normalization	**87.71**	88.09
Word Clustering	86.98	87.47
Semantic File	87.31	85.25

Table 8: Accuracy obtained after applying different sparsity-reduction tricks.

5 Error Analysis

We use the best configurations from the previous section on the training and test data, for gold annotation and an automatically derived one. Table 9 shows the final test results of the two parsers for Persian. In addition to LAS, we also include UAS and LA to facilitate comparisons in the future. Baseline results are included in the table. In the case of MaltParser, after applying new configurations on data, we repeat the third phase of MaltOptimizer in order to find the best feature template for the new training data. It seems that the graph-based parser performs better than transitions-based parsers in general. Despite a high overall parsing accuracy, only 1017 and 922 (33.91% and 30.74%) of sentences in the test set (with 2999 sentences) are parsed without errors by MaltParser and MSTParser, respectively. MaltParser has lower overall accuracy compared to MSTParser, but the number of completely correct parsed sentences for MaltParser is more than MSTParser. In the case of predicted setting, as mentioned in section 4, there are four values for TMA. This means that we cannot create tense and mood from TMA. For this reason, we force to use TMA in the final configuration of both parsers in the predicted setting.

In order to evaluate parsing errors, we use the same approach as (McDonald and Nivre, 2011) to shows a set of linguistic and structural properties of the baseline and our best setting for each parser[11].

Length Factors Figure 2 shows the accuracy relative to the sentence length in test data. Since there are very limited long sentences in our treebank, the parser cannot predict longer sentences correctly. Consequently, the two parsers tend to have lower accuracies for longer sentences. Both parsers have the same performance, but MSTParser tends to perform better on shorter sentences, that is in contrast with results showed by McDonald and Nivre (2011). We compare each parser with its corresponding baselines. Both parsers in all lengths perform better than their baselines. For MaltParser, improvements occur for longer sentences while for MSTParser improvements occur at smaller sentences. These results are in contrast with the results reported by McDonald and Nivre (2011).

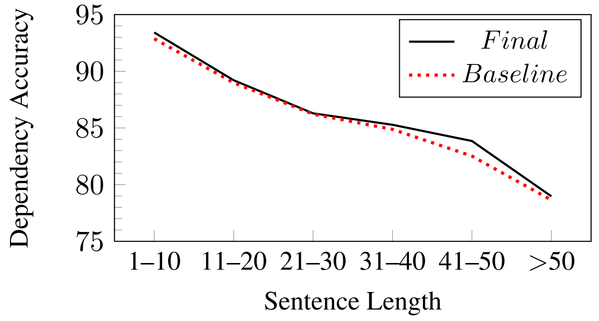
Graph Factors Figure 3 shows the accuracy for arcs relative to their distance to the artificial root node[12]. The area under the curve of final MaltParser

[11]In our analysis, we use MaltEval (Nilsson and Nivre, 2008).

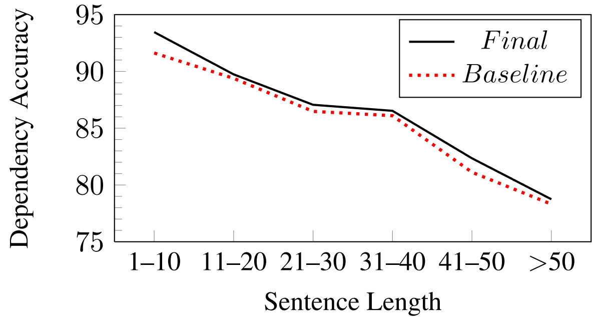
[12]Number of arcs in the reverse path from the modifier of the arc to the root.

Parser	**Method**	**LAS**		**UAS**		**LA**	
Malt	Baseline	87.68	(87.04)	90.41	(89.92)	90.03	(89.49)
	Final	87.91^{++}	(87.16)$^{+}$	90.58^{+}	(90.05)$^{++}$	90.22^{+}	(89.60)$^{+}$
	Diff.	+0.23	(+0.12)	+0.17	(+0.13)	+0.19	(+0.11)
MST	Baseline	87.98	(86.82)	91.30	(90.27)	90.53	(89.90)
	Final	88.37^{++}	(86.97)	91.55^{++}	(90.36)	90.86^{++}	(90.05)
	Diff.	+0.39	(+0.15)	+0.25	(+0.09)	+0.33	(+0.15)

Table 9: Baseline and final results of gold (predicted) test data for MaltParser

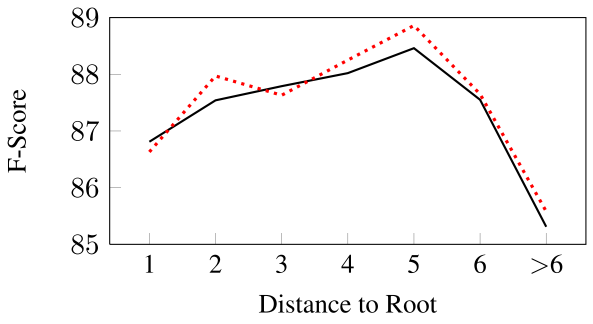


(a) Accuracy of MaltParser per sentence length

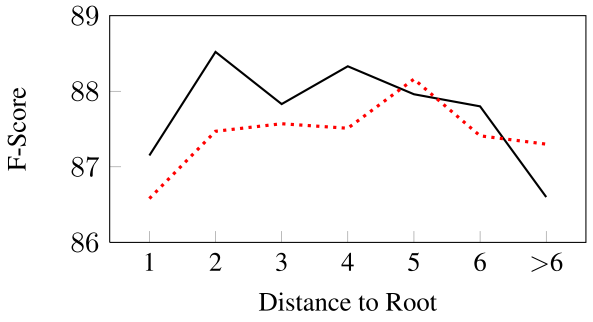


(b) Accuracy of MSTParser per sentence length

Figure 2: Accuracy relative to sentence length. Both parsers perform better than their baselines.



(a) Baseline (⋯⋯) and final (——) accuracy of MaltParser



(b) Baseline (⋯⋯) and final (——) accuracy of MSTParser

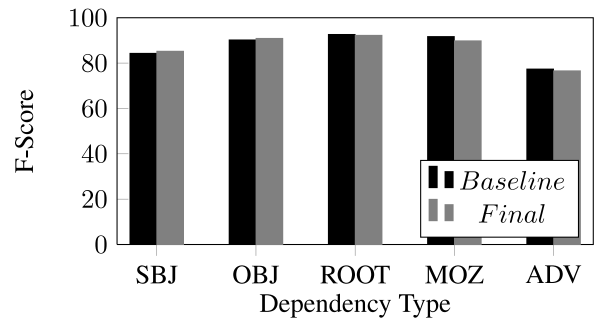
Figure 3: Dependency arc F-score relative to the distance to root

is less than baseline, but it is over baseline for MSTParser. F-score of MSTParser for shorter distance is much better than the baseline and by increasing the distance to root, F-score degrades to be less than the baseline.

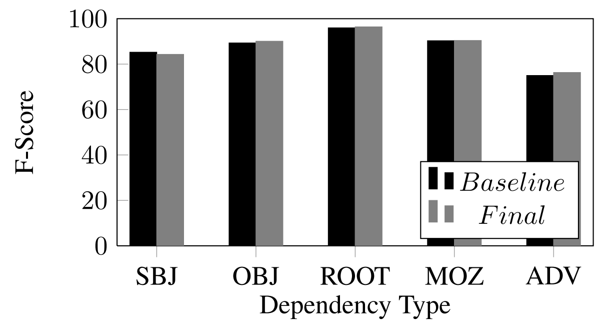
Linguistic Factors MaltParser and MSTParser can find 90.22% and 90.86% of all labels correctly. Figure 4 shows the F-score of some important dependency labels in the test data. MaltParser only improves subject and object categories, while MSTParser improves object, ROOT, and adverb categories. If we only consider the final results, MaltParser performs better for predicting subject and object, while MSTParser performs better for predicting ROOT and ezafe dependent (MOZ)[13], and both have the same accuracy for adverb.

Table 10 gives the accuracy of arcs for each dependent part-of-speech. Final MSTParser performs

[13] *Ezafe construction* is referred to nouns or pronouns that imply a possessed-possessor relation (like first name-last name). The relation between the possessed and possessor is called mozaf (MOZ) that its sign is a vowel /e/ that pronounced right after the head noun (Dadegan Research Group, 2012).



(a) Accuracy of MaltParser per dependency type



(b) Accuracy of MSTParser per dependency type

Figure 4: Dependency label F-score relative to some dependency types.

better than its baseline for all categories, except pronouns and better than MaltParser for all categories, except preposition. Final MaltParser, performs better than its baseline in all categories, except preposition.

6 Conclusion

In this paper, we have investigated a number of issues in data-driven dependency parsing of Persian. Because there is no previous study on parsing the Persian dependency treebank (Rasooli et al., 2013), we first have drawn the baseline for each parser, by selecting best performing algorithm and tuning its parameters. For MaltParser (Nivre et al., 2007b) different between best algorithm (non-projective version of Covington) with default settings and after optimizing feature template by the third phase of MaltOptimizer (Ballesteros and Nivre, 2012) is about 1.5 percent. This shows that the definition of feature template is a crucial aspect of transition-based parsing.

Our first experiment shows the effect of using automatic annotation of POS tags and morphological information. Our new configuration improves two parsers in both gold and predicted setting, but the improvement for MSTParser is higher than for MaltParser. MSTParser has higher accuracy in the gold setting, while MaltParser has better performance in predicted setting. It might mean that MaltParser is more robust against noisy information.

In the second experiment, we have explored the best combination of morphological and lexical semantic features for dependency parsing of Persian. We find that the combination of one morphological feature and one lexical semantic feature gets the best combination for each parser. Our lexical semantic features can be automatically produced for any word and thus we need to predict one morphological feature for real-world settings.

Finally we have proposed two simple methods for reducing data sparsity of each parser. After applying our solutions to three types of challenges, we reached 87.91% and 88.37% LAS on the test set (0.23% and 0.39% improvement over our baseline) for MaltParser and MSTParser, respectively.

Note that all of the experiments we reported in this paper use existing parsers as black boxes. We only changed the input data to obtain the best possible performance given our data sets. We plan to explore modifications of the underlying parsing algorithms to better make use of morphological information.

POS	**Malt**		**MST**	
	Baseline	**Final**	**Baseline**	**Final**
Verb	89.96	90.09	90.96	**91.86**
Noun	89.67	90.13	90.15	**90.23**
Pronoun	92.56	92.94	**93.53**	93.43
Adjective	87.80	88.37	87.77	**88.56**
Adverb	80.80	82.37	82.61	**83.94**
Conjunction	86.03	86.40	86.58	**87.36**
Preposition	**70.93**	70.32	69.74	70.76

Table 10: Accuracy for each dependent part of speech

Acknowledgments

We would like to thank Mohammad-Sadegh Rasooli and our anonymous reviewers for helpful feedback and suggestions. We would also thank Zahra Sarabi

for providing us the data and information about the PLP toolkit.

References

Eneko Agirre, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. 2011. Improving Dependency Parsing with Semantic Classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11): shortpapers*, pages 699–703.

Baharat Ram Ambati, Samar Husain, Sambhav Jain, Dipti Misra Sharma, and Rajeev Sangal. 2010a. Two methods to incorporate local morphosyntactic features in Hindi dependency parsing. In *Proceedings of NAACL HLT 2010 First workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2010)*, pages 22–30.

Baharat Ram Ambati, Samar Husain, Joakim Nivre, and Rajeev Sangal. 2010b. On the Role of Morphosyntactic Features in Hindi Dependency Parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 94–102.

Maryam Aminian, Mohammad Sadegh Rasooli, and Hossein Sameti. 2013. Unsupervised Induction of Persian Semantic Verb Classes Based on Syntactic Information. In *Language Processing and Intelligent Information Systems*, pages 112–124.

Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: A System for MaltParser Optimization. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 23–27.

Miguel Ballesteros and Joakim Nivre. 2013. Going to the Roots of Dependency Parsing. *Computational Linguistics*, pages 5–13.

Kepa Bengoetxea and Koldo Gojenola. 2010. Application of Different Techniques to Dependency Parsing of Basque. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 31–39.

Mahmood Bijankhan. 2004. The role of the corpus in writing a grammar: An introduction to a software. *Iranian Journal of Linguistics*.

Bernd Bohnet and Jonas Kuhn. 2012. The Best of Both Worlds A Graph-based Completion Model for Transition-based Parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87.

Bernd Bohnet and Joakim Nivre. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 1455–1465.

Jinho D. Choi and Martha Palmer. 2011a. Getting the Most out of Transition-based Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11): shortpapers*, pages 687–692.

Jinho D. Choi and Martha Palmer. 2011b. Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing. In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, pages 1–11.

Dadegan Research Group. 2012. Persian Dependency Treebank Annotation Manual and User Guide. Technical report, SCICT.

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*, pages 103–107.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*, pages 595–603.

Percy Liang. 2005. *Semi-Supervised Learning for Natural Language*. Ph.D. thesis, Massachusetts Institute of Technology.

Yuval Marton, Nizar Habash, and Owen Rambow. 2011. Improving Arabic Dependency Parsing with Form-based and Functional Morphological Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11)*, pages 1586–1596.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and Integrating Dependency Parsers. *Computational Linguistics*, pages 197–230.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

Jens Nilsson and Joakim Nivre. 2008. MaltEval: An Evaluation and Visualization Tool for Dependency Parsing. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC '08)*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 99–106.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 Shared Task on Dependency

Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

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

Joakim Nivre. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing (IJCNLP) of the AFNLP*, pages 351–359.

Mohammad Sadegh Rasooli, Omid Kashefi, and Behrouz Minaei-Bidgoli. 2011. Effect of Adaptive Spell Checking in Persian. In *7th International Conference on Natural Language Processing andKnowledge Engineering (NLP-KE)*, pages 161–164.

Mohammad Sadegh Rasooli, Manouchehr Kouhestani, and Amirsaeid Moloodi. 2013. Development of a Persian Syntactic Dependency Treebank. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 306–314.

Zahra Sarabi, Hooman Mahyar, and Mojgan Farhoodi. 2013. PLP Toolkit: Persian Language Processing Toolkit. In *3rd International eConference on Computer and Knowledge Engineering (ICCKE 2013)*.

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and Syntactic Case in Statistical Dependency Parsing. *Computational Linguistics*, pages 23–55.

Mojgan Seraji, Beáta Megyesi, and Joakim Nivre. 2012a. Bootstrapping a Persian Dependency Treebank. *Linguistic Issues in Language Technology*, pages 1–10.

Mojgan Seraji, Beáta Megyesi, and Joakim Nivre. 2012b. Dependency Parsers for Persian. In *Proceedings of 10th Workshop on Asian Language Resources, COLING 2012, 24th International Conference on Computational Linguistics*.

Mehrnoush Shamsfard, Akbar Hesabi, Hakimeh Fadaei, Niloofar Mansoory, Ali Famian, Somayeh Bagherbeigi, Elham Fekri, Maliheh Monshizadeh, and S. Mostafa Assi. 2010. Semi Automatic Development Of FarsNet: The Persian Wordnet. In *Proceedings of 5th Global WordNet Conference (GWA2010)*.

Mehrnoush Shamsfard. 2011. Challenges and Open Problems in Persian Text processing. In *The 5th Language and Technology Conference (LTC 2011)*, pages 65–69.

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