On the Importance of Ezafe Construction in Persian Parsing

Alireza Nourian*, Mohammad Sadegh Rasooli☾ , Mohsen Imany*, and Heshaam Faili□

*Department of Computer Engineering, Iran University of Science and Technlogy, Tehran, Iran
{nourian,m_imany}@comp.iust.ac.ir
☾ Department of Computer Science, Columbia University, New York, NY, USA
rasooli@cs.columbia.edu
□School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
hfaili@ut.ac.ir

Abstract

Ezafe construction is an idiosyncratic phenomenon in the Persian language. It is a good indicator for phrase boundaries and dependency relations but mostly does not appear in the text. In this paper, we show that adding information about Ezafe construction can give 4.6% relative improvement in dependency parsing and 9% relative improvement in shallow parsing. For evaluation purposes, Ezafe tags are manually annotated in the Persian dependency treebank. Furthermore, to be able to conduct experiments on shallow parsing, we develop a dependency to shallow phrase structure convertor based on the Persian dependencies.

1 Introduction

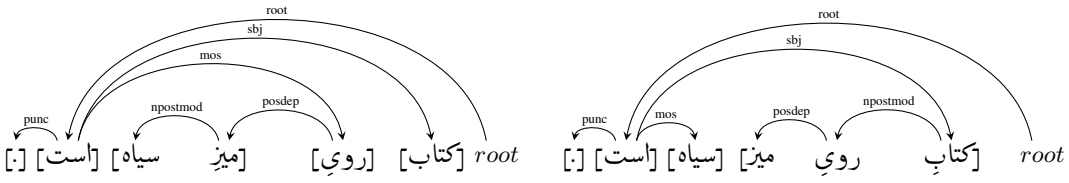
There have been many studies on improving syntactic parsing methods for natural languages. Although most of the parsing methods are language-independent, we may still require some language specific knowledge for improving performance. Besides many studies on parsing morphologically rich languages (Seddah et al., 2013; Seeker and Kuhn, 2013), syntactic parsing for the Persian language is not yet noticeably explored. Concretely speaking, there are some recent work on dependency parsing for Persian (Seraji et al., 2012; Ghayoomi, 2012; Khallash et al., 2013) and very few studies on shallow parsing (Kian et al., 2009).

The main focus of this paper is on the usefulness of Ezafe construction in Persian syntactic processing. Ezafe is an unstressed vowel -e that occurs at the end of some words (-ye in some specific occasions) that links together elements belonging to a single constituent (Ghomeshi, 1997). It often approximately corresponds in usage to the English preposition "of" (Abrahams, 2004). In the following example, the first word has an Ezafe vowel:

$$\begin{cases} montazer_e & Ab \\ waiting_{Ezafe} & water \end{cases} \quad waiting\ for\ water$$

This is an idiosyncratic construction that appears in the Persian language with Perso-Arabic script. This construction is similar to Idafa construction in Arabic and construct state in Hebrew (Habash, 2010). It is mostly used for showing a possessive marker, adjective of a noun or connecting parts of a name (i.e. first and last name) or title. As a general statement, Ezafe occurs between any two items that have some sort of connection (Ghomeshi, 1997). Ezafe vowel is attached to the head noun and to the modifiers that follow it: attributive nouns, adjectival and prepositional phrases (Samvelian, 2006). As depicted in Figure 1, this construction is very useful for disambiguating syntactic structures. The main issue here is that Ezafe rarely appears in the written text. This relies on the fact that Persian is written in Perso-Arabic script and vowels are mostly not written.

There are few studies (Noferesti and Shamsfard, 2014; Asghari et al., 2014) on automatically finding Ezafe construction. In this work, we modify the part of speech tagset for the Persian words. This is done by adding an indicator of Ezafe to each part of speech (POS) tag and then train a supervised tagger on the modified tags. We show that having this modified tagset can both improve dependency parsing and shallow parsing (chunking). We achieve 12.8% and 4.6% relative error reduction in dependency parsing with gold and auto-



(a) First reading: The book is on the black table.

(b) Second reading: The book on the table is black.

Figure 1: This figure shows two different readings for the same sentence with different Ezafe constructions. As shown in the trees, Ezafe affects both phrase boundaries and dependency relations.

matic POS tags. We also achieve 31% and 9% relative error reduction in shallow parsing with gold and automatic POS tags.

Our work is not only restricted to the effect of Ezafe in parsing, but as a byproduct, we create an open-source rule-based dependency to chunk converter for the Persian language. We have also manually tagged all words in the Persian dependency treebank (Rasooli et al., 2013) with 99.6% annotator agreement. This dataset is available for research purposes.[1]

The main contributions of this paper are: 1) showing the usefulness of Ezafe construction on dependency parsing and chunking, 2) developing a statistical chunker for the Persian language, 3) enriching the Persian treebank with manual Ezafe tags. The remainder of this paper is organized as the following: we describe our approach and data preparation in §2 and then conduct experiments in §3. Error analysis and conclusion are made in §4 and §5.

2 Data Preparation

We define a simple procedure to include the information about Ezafe construction in our data. Concretely, we attach the Ezafe indicator to the tags and train a POS tagger on the new tagset. This idea is very similar to that of (Asghari et al., 2014). Thanks to the presence of Ezafe feature in the Peykare corpus (Bijankhan et al., 2011), we can easily train a POS tagger on the new tagset. We use the developed tagger to tag the dependency treebank. Peykare corpus has approximately ten million tokens and can give us a very accurate POS tagger even with the finer-grained Ezafe tags. We try this idea on two different tasks: dependency parsing and shallow parsing.

[1] http://dadegan.ir/catalog/ezafe

2.1 Chunking Data Preparation

Unfortunately there is no standard chunking data for the Persian language. To compensate for this, we define the following rules to convert a dependency tree (based on Dadegan treebank dependencies (Rasooli et al., 2013)) to a shallow phrase structure:

- We initialize every node (word) as a separate chunk; e.g. verb creates a VP.
- If a node has a head or dependent belonging to a chunk (without any gap), attach that node to the same chunk.
- If a node is a preposition/postposition, attach it to its next/previous dependent and create a PP.
- A node with a dependency relation "non-verbal element", "verb particle", or "enclitic non-verbal element" belongs to the same VP as its head.
- If a node is a particle, subordinating clause, coordinating conjunction, or punctuation, we should not create an independent chunk for it.
- If a node is a "noun post modifier" or "Ezafe dependent", attach it to its parent chunk.
- If a node is a "conjunction of a noun" or "conjunction of an adjective" and has a sibling with either "Ezafe dependent" or "noun post-modifier" dependency relation, it should have the same chunk as its parent.
- If a node with pseudo-sentence POS has an adverbial dependency with its parent, it creates an ADVP and otherwise a VP.

Implementation of the above rules is available in the Hazm toolkit.[2] There are some minor exceptions in the above rules that are handled manually in the toolkit.

3 Experiments

In this section we describe our experiments on Ezafe tagging, parsing and also adding manual Ezafe tags to the Persian dependency treebank.

3.1 Automatic Ezafe Tagging

As mentioned in §2, we attach Ezafe feature indicator to the tags and train a POS tagger on the new tagset. We use Wapiti tagger (Lavergne et al., 2010) to train a standard trigram CRF sequence tagger model with standard transition features and the following emission features: word form of the current, previous and next word, combination of the current word and next word, combination of the current word and previous word, prefixes and suffixes up to length 3, indicator of punctuation and number (digit) for the current, previous and next word. The tagger has an accuracy of 98.71% with the original tagset and 97.33% with the modified tagset.

3.2 Gold Standard Ezafe Tags

The Persian dependency treebank does not provide gold Ezafe tags. In order to evaluate the effect of gold Ezafe tags, we try to manually annotate Ezafe in the treebank. This is done by six annotators where all of them are native speakers and linguists. The inter-annotator agreement of a small portion of the data (one thousand sentences) is 99.6%. Our manual investigation shows that almost half of the disagreements was because of the mistakes and not because of the complicated structure. Table 1 shows the statistics about the presence of Ezafe tag for each specific POS.

3.3 Chunking

We use Wapiti tagger (Lavergne et al., 2010) to train a standard CRF tagger with IOB tags for phrase chunking. The features include third order transition features and emission features of word form and POS for the current word, previous word and the word before it, the next word and the word after it. As shown in Table 2 and 3, our intuition holds for both gold and automatic tags. We observe that using Ezafe on gold tags, gives us better performance compared to using coarse-grained POS tags and also fine-grained POS tags (FPOS) provided by the dependency treebank annotators. The tagset in Peykare corpus is very different from the treebank. Because of this inconsistency, we could not reproduce the results with automatic FPOS tags trained on Peykare corpus. Our experiments on training solely on the treebank FPOS tags do not give us a reliable FPOS tagger and this leads to very low parsing accuracy. Therefore we do not conduct experiments with automatic FPOS tags. Table 3 shows the results with automatic tags. As shown in the table, using the the Ezafe tagset improves the chunking accuracy.

[2] https://github.com/sobhe/hazm

Tag	Freq.	Relative Freq.	Ezafe %
N	190048	39.24%	34.22%
PREP	56376	11.64%	12.04%
ADJ	35902	7.41%	17.45%
PRENUM	6018	1.24%	1.21%
IDEN	835	0.17%	5.03%
POSNUM	560	0.12%	30.71%
other	194572	40.18%	00.10%

Table 1: Statistics about Ezafe for each POS tag in the Persian dependency treebank.

Tagset	Precision	Recall	F-Measure
POS	91.98%	90.37%	91.17%
FPOS	92.37%	90.92%	91.64%
POSe	**93.88%**	**93.97%**	**93.92%**

Table 2: Chunking results on the Persian dependency treebank test data with gold POS tags. FPOS refers to the fine-grained POS tags in the Persian dependency treebank and POSe is the modified Ezafe-enriched tagset.

Tagset	Tag Acc.	Precision	Recall	F-Measure
POS	98.71%	89.44%	88.02%	88.72%
POSe	97.33%	**90.42%**	**89.13%**	**89.77%**

Table 3: Chunking results on the Persian dependency treebank test data with automatic POS tags.

3.4 Dependency Parsing

Similar to the chunking experiments, we provide two sets of experiments to validate our hypothesis about the importance of Ezafe construction. We

Tagset	MaltParser		YaraParser		TurboParser	
	LAS	UAS	LAS	UAS	LAS	UAS
POS	88.13%	90.69%	88.60%	91.17%	89.88%	92.25%
FPOS	88.46%	91.01%	89.02%	91.56%	89.98%	92.30%
POSe	**89.12%**	**91.64%**	**89.91%**	**92.42%**	**90.85%**	**93.24%**

Table 4: Dependency Parsing results on the test data with different gold standard tagsets. UAS is the unlabeled attachment score and LAS is the labeled attachment score.

Tagset	Tag acc.	MaltParser		YaraParser		TurboParser	
		LAS	UAS	LAS	UAS	LAS	UAS
POS tagger	98.71%	85.34%	88.80%	85.90%	89.43%	87.28%	90.59%
POSe tagger	97.33%	**85.74%**	**89.24%**	**86.35%**	**89.86%**	**87.73%**	**91.02%**

Table 5: Dependency Parsing results on the test data with different automatic tagsets.

use three different off-the-shelf parsers: 1) Malt parser v1.8 (Nivre et al., 2007), 2) Yara parser v0.2 (Rasooli and Tetreault, 2015), and 3) Turbo parser v2.2 (Martins et al., 2013). We train Malt with Covington non-projective algorithm (Covington, 1990) after optimizing it with Malt optimizer (Ballesteros and Nivre, 2012), Yara with the default settings (64 beam) and 10 training epochs and Turbo with its default settings. The main reason for picking these three parsers is that we want to see the effect of Ezafe construction on a greedy parser (Malt), beam parser (Yara), and a graph-based parser (Turbo). As shown in Table 4 and 5, the parsing accuracy is improved across all different parsers by using the Ezafe tagset.

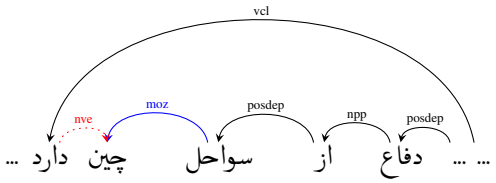
4 Error Analysis

In this section we provide some error analysis for showing the effectiveness of our approach.

Effect on the common POS tags Our investigation on the development data shows that the dependency attachment accuracy is improved by 6.5% for adjectives and 6.2% for nouns. This is consistent with our intuition because Ezafe construction mostly occurs in nouns and adjectives. It is worth noting that for some tags such as determiners the Ezafe construction does not help.

Ezafe indicator as a feature We try to use Ezafe as an independent feature in Malt parser. This is done by adding the indicator in the feature column in CoNLL dependency format. We then use Malt optimizer (Ballesteros and Nivre, 2012) to find the optimized feature setting. We see that adding this feature gives us the same accuracy improvement as having the modified tagset. This shows that we do not really need to have a parser that uses extra features to add Ezafe information.

Manual data investigation We randomly picked some sentences from the development data and observed the same effect as we could expect from adding Ezafe to the tagset: the main gain is on those sentences where the presence/absence of Ezafe construction is crucial for making correct decisions by the parser. For example, in the following sub-sentence, the word چین means "China" but the dependency parser without knowing Ezafe tag, confused it with the other meaning: "ruffle" and created a "non-verbal element" (light verb) dependency with the verb, instead of making it an Ezafe dependent to the previous word (سواحل).



Effect on the training data size For investigating the benefit of Ezafe construction, we train Malt parser on different data sizes starting from 50% of the original size. This trend is depicted in Figure 2. The interesting fact is that we can leverage Ezafe construction and use only 70% of the training data while reaching the accuracy of the original part of speech tagset trained on the whole data.

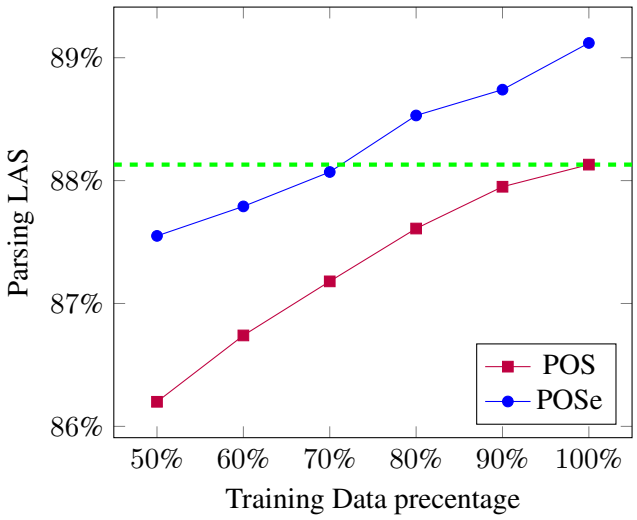


Figure 2: Trend on the data size and accuracy. As shown by the horizontal dashed line, Ezafe tags can improve over standard tags while having approximately 70% of the data.

5 Conclusion

In this paper we showed the effectiveness of Ezafe construction as a robust feature for syntactic parsing in Persian. One interesting direction for further research would be to show the effect of this feature in other natural language processing tasks.

Acknowledgement

We thank Computer Research Center of Islamic Sciences (CRCIS) for supporting us on corpus annotation. We thank Parinaz Dadras, Saeedeh Ghadrdoost-Nakhchi, Manouchehr Kouhestani, Mostafa Mahdavi, Azadeh Mirzaei, Neda Poormorteza-Khameneh, Morteza Rezaei-Sharifabadi and Salimeh Zamani for helping us on annotation and the three anonymous reviewers for their helpful comments.

References

Simin Abrahams. 2004. *Modern Persian: a coursebook*. Routledge.

Habibollah Asghari, Jalal Maleki, and Heshaam Faili. 2014. A probabilistic approach to persian ezafe recognition. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 138–142, Gothenburg, Sweden, April. Association for Computational Linguistics.

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: A system for maltparser optimization. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 2757–2763, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. 2011. Lessons from building a Persian written corpus: Peykare. *Language Resources and Evaluation*, 45:143–164.

Michael A Covington. 1990. Parsing discontinuous constituents in dependency grammar. *Computational Linguistics*, 16(4):234–236.

Masood Ghayoomi. 2012. Word clustering for Persian statistical parsing. In *Advances in Natural Language Processing*, pages 126–137. Springer.

Jila Ghomeshi. 1997. Non-projecting nouns and the ezafe: Construction in Persian. *Natural Language & Linguistic Theory*, 15(4):729–788.

Nizar Y Habash. 2010. Introduction to Arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Mojtaba Khallash, Ali Hadian, and Behrouz Minaei-Bidgoli. 2013. An empirical study on the effect of morphological and lexical features in Persian dependency parsing. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 97–107, Seattle, Washington, USA, October. Association for Computational Linguistics.

Soheila Kian, Tara Akhavan, and Mehrnoush Shamsfard. 2009. Developing a Persian chunker using a hybrid approach. In *International Multiconference on Computer Science and Information Technology, 2009. IMCSIT'09*, pages 227–234. IEEE.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July.

André F. T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. pages 617–622.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Samira Noferesti and Mehrnoush Shamsfard. 2014. A hybrid algorithm for recognizing the position of ezafe constructions in Persian texts. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2(6):17–25.

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

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.

Pollet Samvelian. 2006. When morphology does better than syntax: the Ezafe construction in Persian. *Ms., Université de Paris*, (1997):1–54.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richard Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Wolinski, Alina Wroblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task : Cross-framework evaluation of parsing morphologically rich languages. *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*, (October):146–182.

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and syntactic case in statistical dependency parsing. *Computational Linguistics*, 39(1):23–55.

Mojgan Seraji, Beáta Megyesi, and Joakim Nivre. 2012. Dependency parsers for Persian. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 35–44, Mumbai, India, December. The COLING 2012 Organizing Committee.