The CMU METAL Farsi NLP Approach

Weston Feely[†] **Mehdi Manshadi**[*] **Robert Frederking**[†] **Lori Levin**[†]
[†]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA
{wfeely, ref, lsl}@cs.cmu.edu
[*]Department of Computer Science, University of Rochester, Rochester, NY
mehdih@cs.rochester.edu

Abstract

While many high-quality tools are available for analyzing major languages such as English, equivalent freely-available tools for important but lower-resourced languages such as Farsi are more difficult to acquire and integrate into a useful NLP front end. We report here on an accurate and efficient Farsi analysis front end that we have assembled, which may be useful to others who wish to work with written Farsi. The pre-existing components and resources that we incorporated include the Carnegie Mellon TurboParser and TurboTagger (Martins et al., 2010) trained on the Dadegan Treebank (Rasooli et al., 2013), the Uppsala Farsi text normalizer PrePer (Seraji, 2013), the Uppsala Farsi tokenizer (Seraji et al., 2012a), and Jon Dehdari's PerStem (Jadidinejad et al., 2010). This set of tools (combined with additional normalization and tokenization modules that we have developed and made available) achieves a dependency parsing labeled attachment score of 89.49%, unlabeled attachment score of 92.19%, and label accuracy score of 91.38% on a held-out parsing test data set. All of the components and resources used are freely available. In addition to describing the components and resources, we also explain the rationale for our choices.

Keywords: Farsi (Persian), Natural Language Processing, Dependency Parsing

1. Introduction

As part of the Carnegie Mellon University METAL multi-lingual metaphor understanding project, we needed to analyze large corpora of written Farsi sentences. Since this was one small part of a large project, we did not have sufficient internal resources to develop significant Farsi tools ourselves. We therefore sought out freely available Farsi resources, combined them, compared components when more than one was available and developed a few missing pieces ourselves. This resulted in a set of tools with dependency parsing labeled attachment score of 89.49%, unlabeled attachment score of 92.19%, and label accuracy score of 91.38% on a held-out parsing test data set. In the overall metaphor detection system, the results of the dependency parser were used as input to our Conventionalized Conceptual Metaphor (CCM) detector (Levin et al., 2014), as well as other detectors.

We must emphasize that most of the tools described here were developed by others, but by selecting and combining these tools with the additional small tools that we developed, we have produced an accurate and efficient Farsi analysis front end that may be useful to others who wish to work with written Farsi. All the components are freely available.

We describe below the Farsi resources that we are using, along with the rationale for our choices. To avoid suspense: we are using the Carnegie Mellon TurboParser and TurboTagger (Martins et al., 2010) trained on the Dadegan Treebank (Rasooli et al., 2013), with both the Uppsala Farsi text normalizer PrePer (Seraji, 2013) and our CMU text normalizer (Feely, 2013), the Uppsala Farsi tokenizer (Seraji et al., 2012a), our CMU Farsi verb tokenizer (Manshadi, 2013) and Jon Dehdari's PerStem (Jadidinejad et al., 2010).

2. Farsi Treebank

As the primary data resource for developing our system, we chose the Dadegan Persian Dependency Treebank (Rasooli et al., 2013), a 29,982 sentence dependency-parsed treebank, which we currently use to train our Farsi TurboParser and TurboTagger. The Dadegan treebank was developed mostly on newswire text and uses its own part-of-speech formalism and dependency label formalism. We obtain very accurate parses and tags using this treebank, when tested on a subset of the treebank's sentences (see numbers below). Notably, this treebank's tokenization scheme combines verbs together with their affixes and auxiliaries, which poses a difficult NLP task for our Farsi tokenizer (Manshadi, 2013) (see below). To explain with a parallel English example, this is like treating "should have tried" as a single English token, rather than as two auxiliaries and an inflected form of the verb "try". But due to its large size, we decided to use the treebank and cope with this idiosyncrasy.

3. Farsi processing components

3.1. Farsi Parser

The dependency parser we use is the Carnegie Mellon TurboParser (Martins et al., 2010), a parsing toolkit created by Noah Smith's group here at Carnegie Mellon, which allows for training a dependency parser on any CONLL-format treebank. We have used this toolkit extensively in the METAL project to train several iterations of our Farsi parser (as well as using it for Spanish and English), producing increasingly accurate results as we obtain more data and adjust the training settings of the parser. We split the Dadegan treebank's 29,982 sentences into a randomly selected 90% of the sentences for training and 10% of the sentences for testing. Our final accuracy on the randomly selected held-out test set, from the Dadegan treebank:

Score Type	Score
Labeled attachment score	39669 / 44327 = 89.49 %
Unlabeled attachment score	40866 / 44327 = 92.19 %
Label accuracy score	40507 / 44327 = 91.38 %

Table 1: Our Farsi Dependency Parser Scores

The parsing scores above are the standard metrics for evaluating dependency parsers, which are automatically created by the evaluation script provided by TurboParser (Martins et al., 2010). Our scores are based on the gold standard treebank dependencies from our randomly-selected held-out test set from the Dadegan Treebank (Rasooli et al., 2013).

For POS tagging, we used TurboTagger, the part-of-speech tagger toolkit that comes with TurboParser. We trained our Farsi tagger on the same training set as our parser, and tested the tagger on the same test set as our parser. TurboTagger provides very accurate POS tagging, with Tagging Accuracy of 95.6% on our randomly selected held-out test set.

3.2. Other Farsi NLP Components

For training, we applied the Uppsala Farsi text pre-processing tools provided by Uppsala University (Seraji, 2013) on the Dadegan treebank, to normalize spacing between Farsi words and their affixes. We then used our own Farsi text normalizer (Feely, 2013), which removes Arabic and Persian diacritics and normalizes variant forms of the Farsi letter "ye" to a single unicode representation. We also used PerStem (Jadidinejad et al., 2010) on the tokens in the entire treebank, to produce a new set of lemmas for the Dadegan treebank. This normalization produces input suitable for the TurboTagger and TurboParser described above.

The following figures are an example of a single Farsi sentence as it passes through each step of our NLP pipeline. In the processing of new Farsi text, we apply the tools in the following order: Uppsala normalizer, CMU normalizer, Uppsala tokenizer, CMU verb tokenizer, PerStem, and then TurboTagger and TurboParser.

سیاست ها کمر تولید را شکسته است.

Figure 1: Original Farsi Sentence

This is the original text for our Farsi sentence. It has not been processed in any way, save for the sentence segmentation that extracted this sentence from its original document. The translation and gloss for this sentence is:

siyâsat	hâ	kamar-e	towlid	râ	šekaste	ast.
policy	PL	back-GEN	production	ACC	broken	has.

Table 2: Gloss of Farsi example sentence
"The policies have broken the back of the production."

The following figures show how this sentence is processed by the different components of our NLP pipeline.

سیاست‌ها کمر تولید را شکسته است.

Figure 2: Text Normalized Farsi Sentence

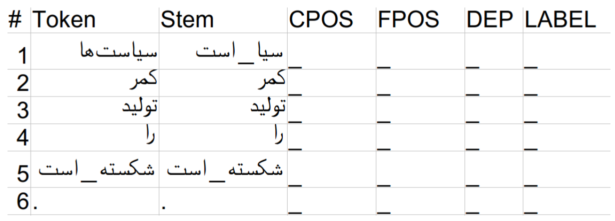
In the above figure 2, we have run both the Uppsala text normalizer PrePer (Seraji, 2013) and our own Farsi text normalizer (Feely, 2013) on the sentence. The Uppsala text normalizer maps any Arabic-specific characters to their Persian unicode equivalents, changes Western digits to their Persian equivalents, and most importantly normalizes the space characters appropriately, so that full spaces between a words and its affix are changed to zero-width-non-joiner characters (ZWNJ) (Seraji, 2013). In this particular sentence, the space between the word for policy "siyâsat" and the following plural marker "hâ" is normalized by PrePer into a ZWNJ character. Our own normalizer would additionally remove diacritics from the sentence, which aids in producing an accurate parse in our final step, but since this sentence has no such diacritics, the normalization effects in this sentence come solely from PrePer.

سیاست‌ها کمر تولید را شکسته_است .

Figure 3: Tokenized Farsi Sentence

In the above figure 3, the same Farsi sentence has now been tokenized using the Uppsala Farsi tokenizer (Seraji et al., 2012a), to split the punctuation off of the words in the sentence. Then, our Farsi verb tokenizer (Manshadi, 2013) has been applied to join the main verb in the sentence (broken "šekaste") with its surrounding affixes and auxiliaries (has "ast") using underscores, in order to form a single token (has broken "šekaste ast"). This aids in our later parsing step, by matching the tokenization style of the Dadegan Treebank (Rasooli et al., 2013).

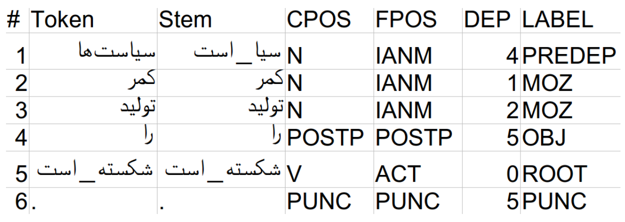
In the below Figure 4, the sentence has been re-formatted to have one word per line, in CONLL-format to prepare for dependency parsing. The words are numbered in the first



#	Token	Stem	CPOS	FPOS	DEP	LABEL
1	سیاست‌ها	سیا_است	_	_	_	_
2	کمر	کمر	_	_	_	_
3	تولید	تولید	_	_	_	_
4	را	را	_	_	_	_
5	شکسته_است	شکسته_است	_	_	_	_
6	.	.	_	_	_	_

Figure 4: Stemmed Farsi Sentence

column, so the tokens are placed in the second column. Additionally, PerStem (Jadidinejad et al., 2010) has been run on each word and the corresponding stem for each word is placed in the following column three. Note that the stem for the first word (policies "siyâsathâ") is incorrect.



#	Token	Stem	CPOS	FPOS	DEP	LABEL
1	سیاست‌ها	سیا_است	N	IANM	4	PREDEP
2	کمر	کمر	N	IANM	1	MOZ
3	تولید	تولید	N	IANM	2	MOZ
4	را	را	POSTP	POSTP	5	OBJ
5	شکسته_است	شکسته_است	V	ACT	0	ROOT
6	.	.	PUNC	PUNC	5	PUNC

Figure 5: Parsed Farsi Sentence

The above Figure 5 shows the final output of our system, after TurboTagger and TurboParser (Martins et al., 2010) have been run on our sentence. The Farsi part-of-speech tags produced by TurboTagger are now in the fourth and fifth columns of the CONLL-format parse. Our part-of-speech tagger produces the fine-grained part-of-speech tags (e.g. "IANM" for inanimate noun) in the fifth column, which we deterministically map back to a set of coarse-grained part-of-speech tags (e.g. "N" for noun) which we put into the fourth column. These coarse-grained part-of-speech tags help us attain our best parsing accuracies. Our part-of-speech tag set comes from the Dadegan Persian Dependency Treebank (Rasooli et al., 2013). You can find documentation for these part-of-speech tags in the treebank's documentation.

The syntactic dependencies produced by TurboParser are in the sixth column and the labels for these dependencies are in the seventh column. Again, the set of syntactic dependency labels comes from the Dadegan Persian Dependency Treebank (Rasooli et al., 2013). You can find documentation for this dependency label set in the treebank's documentation. Note that the parse for this sentence is incorrect; our parser is not perfect and notably has trouble with noun-noun compounds in Farsi, which occur often in sentences like this.

We will now describe each of our NLP tools in some more detail.

3.2.1. Uppsala Farsi Pre-processing tools

Includes a sentence segmenter and a text normalizer (Seraji, 2013), and a Farsi tokenizer (Seraji et al., 2012a). We use the Uppsala text normalizer to normalize Farsi whitespace characters and their Farsi tokenizer to normalize Farsi nouns and adjectives. We use the Uppsala text normalizer together with our own text normalizer, because the combination of the two produced the best parsing results and the most consistent text normalization scheme between the Dadegan treebank and new data.

3.2.2. CMU Farsi text normalizer

A Python script created by our CMU team, to normalize Farsi text by removing diacritics (which are helpful to human readers, but only complicate and confuse automatic processing) and adjusting character variants to their most common forms, to allow our other NLP tools to work on less sparse tokens. This step is applied to our Dadegan treebank before training TurboParser, as well as to new data, to ensure consistency in training and testing data. It is available online (Feely, 2013). We use the Uppsala text normalizer together with our own text normalizer, because the combination of the two produced the best parsing results and the most consistent text normalization scheme between the Dadegan treebank and new data.

3.2.3. CMU Farsi Verbal Morphology Tokenizer

A tokenizer created by one of our Farsi team consultants, designed to group together Farsi verbs with their surrounding affixes and auxiliaries into a single token (Manshadi, 2013). This is a necessary pre-processing step for raw text, due to the nature of the Dadegan treebank, which keeps verbs together with their affixes and auxiliaries, as described above. To ensure an accurate parse, we apply this tokenizer as well as the Uppsala tokenizer, which accurately tokenizes Farsi nouns and adjectives when parsing raw text. We tested the accuracy of our verbal morphology tokenizer by using it to reproduce the verb tokenization of the randomly selected held-out parsing test set we chose from the Dadegan treebank, using a de-tokenized version of the treebank's test set sentences. Our tool recreates the exact verb tokenization of the Dadegan treebank, when applied to this detokenized test set (100% precision on our test set). This tool is no longer under active development, but it accounts for the majority of the possible tense, aspect, and moods for Farsi verbs (including many of those not present in our test set).

3.2.4. PerStem Stemmer

A freely-available Farsi stemmer created by Jon Dehdari (Jadidinejad et al., 2010). We currently use this to create Farsi lemmas from our tokens, since the "dictionary" form of verbs is required by our down-stream metaphor detection pipelines (CCM, etc.) The current version of PerStem does not perform very well producing lemmas for our Farsi tokens. It was our hope to replace this stemmer with a more accurate Farsi lemmatizer of our own design, but this work is no longer under active development.

4. Future Work

Our CMU Farsi verbal morphology tokenizer is no longer under active development, but we have accounted for a majority of the possible Farsi verb forms, in order to ensure accurate parses for new texts. One further experiment using this work would be to retrain our parser on the Uppsala Persian Dependency treebank (Seraji et al., 2012b), once the size of the Uppsala treebank is large enough to match the accuracy of our parser trained on the Dadegan treebank. This resource is described below:

4.1. Uppsala Persian dependency treebank (UPDT)

A 6,000 sentence dependency parsed treebank created by Uppsala University (Seraji et al., 2012b), suggested to us recently by our sponsor. This treebank is used to train the freely-available Uppsala Farsi parser (Seraji et al., 2012c), which has a dependency parsing labeled attachment score of 68.68%, unlabeled attachment score of 74.81%, and label accuracy score of 80.64% (compare to our results above). Although much smaller than the Dadegan treebank (Rasooli et al., 2013), this treebank has the benefit of not grouping together verbs and their auxiliaries into a single token. This would reduce the burden on our Farsi tokenizer to produce complex verb tokens, but the treebank's small size so far does not yet allow it to produce better parsing performance. A planned future direction for Farsi is to see whether we can apply this better scheme to the much-larger Dadegan Treebank; alternatively, a larger Uppsala treebank may be released eventually.

5. Conclusion

In conclusion, our Farsi text pre-processing, part-of-speech tagging, and dependency parsing approach produces highly accurate tags and parses on our randomly selected held-out test data set. We have supplemented available Farsi resources by creating the additional text processing tools (our text normalizer and verbal morphology tokenizer) necessary to use these tools to produce accurate parses on new Farsi texts. All of our tools are freely available online, in addition to the previously freely available tools that we used to create this Farsi front-end. It is our hope that this natural language processing front end will be used in future projects, whether related to metaphor detection or other natural language processing tasks.

6. Acknowledgements

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. References

Weston Feely. (2013). Open-source dependency parser, part-of-speech tagger, and text normalizer for Farsi (Persian). https://github.com/wfeely/farsiNLPTools. Accessed 03-2014.

Amir Hossein Jadidinejad, Fariborz Mahmoudi, and Jon Dehdari. (2010). Evaluation of Perstem: A Simple and Efficient Stemming Algorithm for Persian. In Peters, C., Nunzio, G. D., Kurimo, M., Mandl, T., Mostefa, D., Peas, A., and Roda, G., editors, *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, volume 6241 of Lecture Notes in Computer Science, pages 98101. Springer, Heidelberg.

Lori Levin, Teruko Mitamura, Brian MacWhinney, Davida Fromm, Jaime Carbonell, Weston Feely, Robert Frederking, Anatole Gershman and Carlos Ramirez. (2014). Resources for the Detection of Conventionalized Metaphors in Four Languages. In *Proceedings of the Ninth Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Mehdi Manshadi. (2013). Farsi Verb Tokenizer. https://github.com/mehdi-manshadi/Farsi-Verb-Tokenizer. Accessed 03-2014.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. (2010). TurboParsers: Dependency Parsing by Approximate Variational Inference. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

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 (NAACL HLT)*, Atlanta, GA, USA.

Mojgan Seraji, Beáta Megyesi, Joakim Nivre. (2012). A Basic Language Resource Kit for Persian. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey.

Mojgan Seraji, Beáta Megyesi, Joakim Nivre. (2012). Bootstrapping a Persian Dependency Treebank. Published as a Journal in *Special Issue of the Linguistic Issues in Language Technology (LiLT)*, Heidelberg, Germany.

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

Mojgan Seraji. (2013). PrePer: A Pre-processor for Persian. In *Proceedings of The Fifth International Conference on Iranian Linguistics (ICIL5)*, Bamberg, Germany.