From Grammar Rule Extraction to Treebanking: A Bootstrapping Approach

Masood Ghayoomi

German Grammar Group
Freie Universität Berlin, Germany
masood.ghayoomi@fu-berlin.de

Abstract

Most of the reliable language resources are developed via human supervision. Developing supervised annotated data is hard and tedious, and it will be very time consuming when it is done totally manually; as a result, various types of annotated data, including treebanks, are not available for many languages. Considering that a portion of the language is regular, we can define regular expressions as grammar rules to recognize the strings which match the regular expressions, and reduce the human effort to annotate further unseen data. In this paper, we propose an incremental bootstrapping approach via extracting grammar rules when no treebank is available in the first step. Since Persian suffers from lack of available data sources, we have applied our method to develop a treebank for this language. Our experiment shows that this approach significantly decreases the amount of manual effort in the annotation process while enlarging the treebank.

Keywords: Treebank Development, Bootstrapping Approach, Grammar Rule Extraction, the Persian Language

1. Introduction

Supervised grammar induction is the task of learning a formal grammar from the observed data and building a model based on this data. Generally, grammar induction is a search space problem and stochastic modeling is widely used to propose a model as a hypothesis for unseen data from the trained, observed data. The input data to build the model and to learn the grammar is a corpus constructed of syntactically annotated sentences like a treebank.

Using the annotated data of a treebank is a great help to induce the grammar of a language with high confidence. Such a rich data source, however, is not available for many languages, and its development is very expensive and time consuming. As a result, a supervised grammar induction will face a barrier for languages with less developed data sources.

Persian, which is an Indo-European language, is less developed in terms of availability of data sources including a treebank. Having no already developed and available treebank results in not being able to create statistical parsers or evaluate against a gold standard. To develop a treebank for Persian, we propose a grammar induction method which extracts the grammar rules from the annotated data to annotate further unseen data in a condition that there is no available treebank in the first step to build the model[1]. In this method, a bootstrapping algorithm helps the process of treebank development in parallel with a component which extracts the grammar rules from the annotate data. In other words, extracting the grammar rules and treebanking feed each other via a bootstrapping process. The challenge of this approach in contrast to previous studies on grammar induction is that there is no already developed treebank in the first step.

This paper is organized as follows: we express our contribution to develop a treebank for Persian in Section 2. In Section 3, we describe the data source and the annotation tool used in our study. Section 4 brings up the challenge how we can extract grammar rules when there is no already developed treebank. Section 5 is devoted to the algorithm for the bootstrapping approach used in the development of the treebank. Experimental results are reported in Section 6; and the paper is summarized in Section 7.

2. HPSG-based Treebank for Persian

Treebank development can be theory independent or dependent on a linguistic theory. King and Simov (1998), Simov (2001; 2002) have shown a finite theory defined as a set of feature graphs is suitable for the representation of HPSG (Pollard and Sag, 1994). The treebank developed based on this assumption will not have feature structures similar to the normal HPSG, but it is composed of phrase structure trees such that the trees are enriched with basic properties of HPSG such as morphosyntactic lexical knowledge, structure sharing, sort hierarchy, dependency relations with schemas (subject, complement, or adjunct), and binding off the slashed elements.

The recent available methodologies for Persian HPSG at both theoretical and system development levels (Taghvaipour, 2005; Samvelian, 2007; Samvelian and Tseng, 2010; Müller, 2010; Müller and Ghayoomi, 2010; Müller et al., In Preparation) motivated us to select HPSG as the backbone of our treebank. Figures 1 and 2 show two tree representations from our developed HPSG-based treebank for sentences (1) and (2) respectively[2].

(1) بورن به اتفاق یکی از اقوام ش به شهر پراگ مهاجرت کرد .

born	be	ettefāqe	yeki	?az	aqvām
Born	to	company.EZ	one.INDEF	from	relatives
aš	be	šahre	perāg	mohājerat	kard.
CL.3SG	to	city.EZ	Prague	move	did.3SG

‘Born moved to the city of Prague in company with one of his relatives.’

[1]The annotated data set is freely accessible from this URL: http://hpsg.fu-berlin.de/∼ghayoomi/PTB.html

[2]Since Persian is a right-to-left language, the trees in Figures 1 and 2 should be read right-to-left.

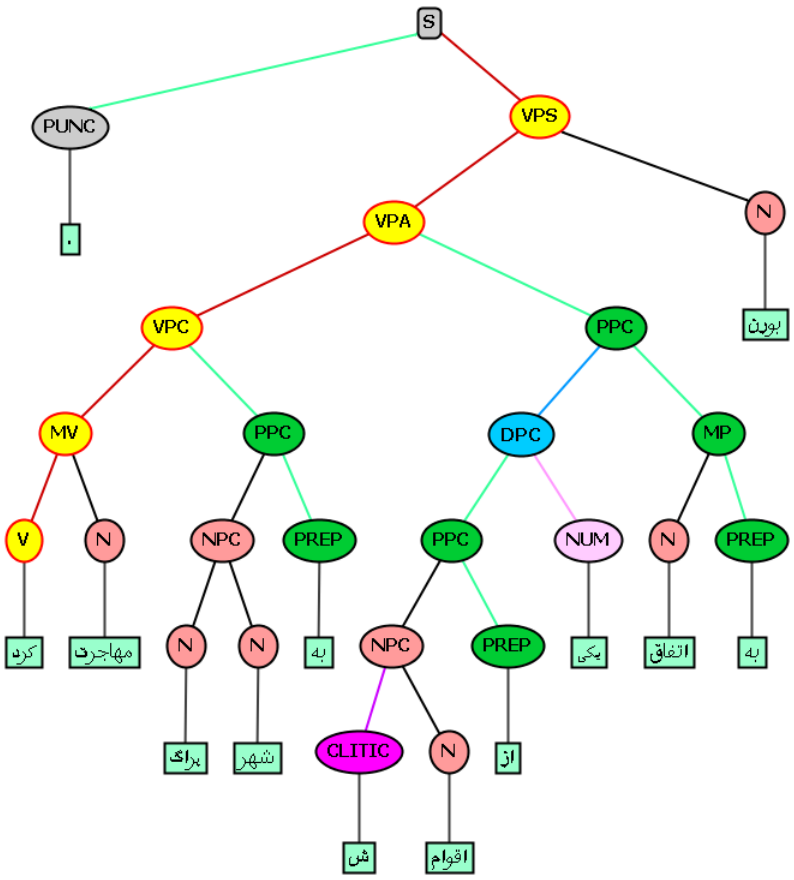


Figure 1: Tree representation of example (1)

(2) نتایج مربوط به تحقیقی است که در سال ۱۹۶۳ انجام شده‌است.

natāyej	marbud	be	tahqiqi	?ast	ke	dar
results	related	to	investigation.REST	is	that	in

sāle	1963	?amjām	šode	?ast.
year.EZ	1963	perform	become	is

‘The results are related to an investigation that was done in the year 1963.’

In the annotation process of our treebank, we used a modified version of the annotation scheme utilized in BulTreeBank, an HPSG-based treebank for Bulgarian (Simov, 2003; Osenova and Simov, 2003). In the annotation, the dependency relations are defined explicitly with these labels: *A* for head-adjunct relation; *C* for head-complement relation; *F* for head-filler relation; *S* for head-subject relation; *SD* for head-subject-drop relation. The extraposed elements are labeled with *DiscE* (Discontinuous Extraposition); and the scrambled elements are labeled with *DiscA* (Discontinuous Adjunct). *nid* (not immediate dominance) defines the canonical position of the extraposed or scrambled elements; and co-referential IDs are used to link *nids* to the extraposed or scrambled elements.

In Figures 1 and 2, the dependency relations are expressed explicitly. Figure 1 is a simple tree in which all the linguistic elements are in their canonical positions; while Figure 2 is a complex tree since a relative clause is extraposed to a non-local position.

3. The Data Source and the Annotation Tool

The data used for our study consists of the 1000 first sentences of the Bijankhan Corpus[3]. The Bijankhan Corpus is a sub-corpus of Peykare, a big balanced corpus for Persian (Bijankhan, 2004; Bijankhan et al., 2011). The Bijankhan Corpus contains more than 2.5 million word tokens, and it is part-of-speech (POS) tagged manually with a rich set of 586 tags. Following the EAGLES guidelines (Leech and Wilson, 1999), there is a hierarchy on the assigned tags such that the first tag expresses the main syntactic category of a word followed by a set of morphosyntactic and semantic features. Mohseni (2008) and Mohseni and Minaei-bidgoli (2010) developed a morphological analyzer for automatic POS tagging and reduced the tag set to 105 tags. The accuracy of the morphological analyzer for inflected words was 95.1%. In our treebank, we used the modified version of the Bijankhan Corpus with the reduced tag set.

Because of two problems in the formatting of the POS tags in the Bijankhan Corpus, even in the modified version, we converted the tags into the MulText-East[4] framework: (1)

[3]http://ece.ut.ac.ir/dbrg/bijankhan/

[4]http://nl.ijs.si/ME/

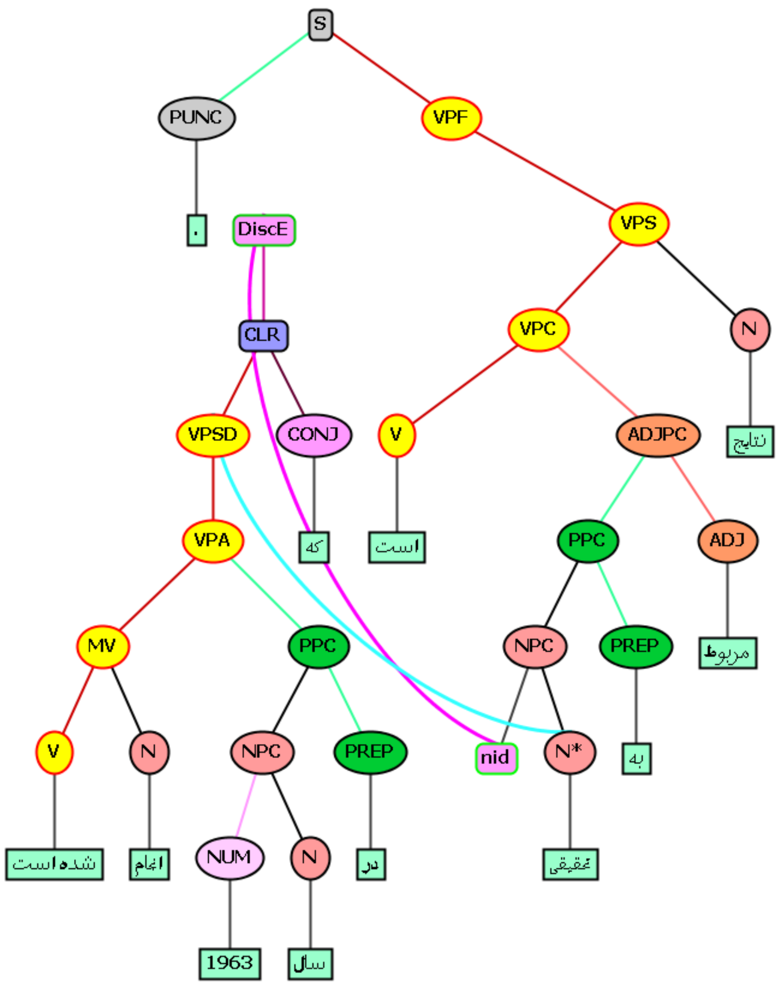


Figure 2: Tree representation of example (2)

the length of the tags with respect to each syntactic category varies; (2) the position of a certain information is not fixed. Table 1 shows examples for the conversion. The information in the MulText-East framework is compressed. Appendix 1 decompresses the information of the tags used in Table 1.

Table 1: Conversion of original POS tags in the Bijankhan Corpus into the MulText-East framework

Persian word	Latin transliteration	English meaning	Original POS tag	Converted POS tag
دفتر	daftar	notebook	N,COM,SING	Ncsp− − −
دفتر	daftar-e	notebook of	N,COM,SING,EZ	Ncsp−−z
دفتر	daftar	office	N,COM,SING,LOC	Ncspk−−
دفتر	daftar-e	office of	N,COM,SING,LOC,EZ	Ncspk−z

To develop the treebank, we benefited from the CLaRK system. CLaRK is an XML-based system for data annotation and corpus development (Simov et al., 2001). Grammar rules are defined as regular expressions in the system manually. Each regular expression is compiled automatically into a deterministic finite state automaton in CLaRK to apply the grammar rules to a document. Since the grammar rules should be applied to XML documents, they should be rewritten as queries with the XPath language. DTDs in XML documents define a document structure with a list of legal elements and attributes. DTDs play several roles in the CLaRK system: (1) they represent the sort hierarchy similar to TRALE (Penn, 2004); (2) they function as a constraint on dominance schemas in HPSG; (3) they monitor and check the consistency during the annotation process.

The CLaRK system supports the cascaded regular grammar introduced by Abney (1996). As a result, the grammar rules in the system will recognize only a portion of a string and not the whole, and the product of one grammar rule is the input to another grammar rule. The result of the cascaded regular grammar is a hierarchical ordering of the grammar rules. In CLaRK, the hierarchy of the grammar rules is constructed manually.

There are several reasons that we decided to use the CLaRK system for our data annotation task: (1) since, to the best knowledge of the author, there is no accessible treebank for Persian, the data annotation process should be started from scratch; as a result, we required a tool that provides us an environment to define regular grammar rules to ease and reduce the manual annotation task; (2) the tool supports Unicode, therefore it nicely handles the Persian script, which is a modified version of the Arabic script; (3) since the backbone of the BulTreeBank (Simov et al., 2003) is similar to

Table 2: Bigram samples for multi tokens written as regular expression in CLaRK

Pattern	**Absolute Freq.**	**Sample**	**Absolute Freq.**	**Translation**	**RE in CLaRK**
E– Ncsp––z	184283	به اتفاق	118	with	**Left RE** **RE:** <“ به ”>,<“ اتفاق ”> **Right RE** **Return Markup:** <MP r=“1-001”> \ w< /MP>
Ncsp– – – Vpyssht– – ––	39213	مهاجرت کرد	10	moved	**Left RE** **RE:** <“Ncsp– – –”>,<“V@ys@@t– – ––”> **Right RE** **Return Markup:** <MV r=“1-001”> \ w< /MV>
Ncsp– – – Vpyssh–f– – –	12011	انجام شده‌است	138	was done	**Left RE** **RE:** <“Ncsp– – –”>,<“V@ys@@–f– – –”> **Right RE** **Return Markup:** <MV r=“1-002”> \ w< /MV>

Table 3: Bigram samples for phrasal constituents written as regular expression in CLaRK

Pattern	**Absolute Freq.**	**Sample**	**Absolute Freq.**	**Translation**	**RE in CLaRK**
Ncspk–z Naspk– – –	11260	شهر پراگ	1	the city of Prague	**Left RE** **RE:** <“Ncspk–z”>,<“Naspk– – –”> **Right RE** **Return Markup:** <NPC r=“1-001”> \ w< /NPC>
Ncspt–z Urns– – –	6976	سال ۱۹۶۳	11	the year 1963	**Left RE** **RE:** <“Ncspt–z”>,<“Urns– – –”> **Right RE** **Return Markup:** <NPC r=“1-002”> \ w< /NPC>
Nclp– – – Cez––nshc	1910	اقوام ش	1	his relatives	**Left RE** **RE:** <“Nc@p– – –”>,<“Cez––nshc”> **Right RE** **Return Markup:** <NPC r=“1-003”> \ w< /NPC>

our treebank, it was a strong motivation to use this tool for our task; (4) the data structure of the system's output is an XML document, as a result it makes the annotation flexible to add several linguistic knowledge including named entities, lemmas, and verb stems to the words in addition to the POS tags.

4. Grammar Rule Extraction for Treebanking

Recalling from the introduction, stochastic modeling is frequently used in supervised grammar induction. From the previous section we found out that the CLaRK system does not support stochastic modeling for the cascaded regular grammars, and defining the regular grammars and their hierarchical ordering are done manually. But the mentioned reasons in the previous section persuaded us to use this system to develop a treebank for Persian. To profit more from the CLaRK system, to ease the annotation task, and to decrease the human intervention in developing the data source, we introduced a non-stochastic grammar induction approach to learn the grammar rules from the Persian annotated data in order to annotate further unseen data.

In our view, grammar induction is a task that recognizes the grammar G of a sentence S in a language L. This grammar is composed of a set of grammar rules R ($r \in R$), which is manually defined in CLaRK and organized in a hierarchical order, to analyze unseen data. Compiling the grammar rules R in the system, each rule r is transformed into a deterministic automaton within the system automatically. Giving an unseen sentence S' to the system, if the local condition defined for r is satisfied, then the grammar rule r is applicable on S'. The result is a robust analysis for sequences that satisfy the local conditions.

In the very beginning step to annotate the set of sentences from the Bijankhan Corpus, we needed to define very basic grammar rules in CLaRK to shallow process the sentences. For initialization, basic grammar rules with high coverage are defined in the system. Since we want to have binary branching in the trees and define the relations between the elements, either adjunct or complement, we extracted bigrams from the Bijankhan Corpus. To this aim, bigrams of the POS tags only, and the words with their corresponding POS tags are extracted from the whole corpus, and the most frequent sequences are defined as regular grammars in the system.

Checking the data from bigrams, we defined a number of regular expressions to handle multi tokens in the preprocessing step such as compound prepositions and compound verbs. Then, we defined another set of regular expressions to recognize the most frequent phrasal constituents and to initialize our data annotation task. Table 2 displays sample bigrams for multi tokens, and Table 3 shows sample bigrams for frequent phrasal constituents with their corresponding absolute frequencies from the Bijankhan Corpus written as regular expressions in CLaRK[5]. As shown in the tables, the available morphosyntactic and semantic information of words in their POS tags is used to define the grammar rules as regular expressions. It is possible to write productive grammar rules with a sequence

[5]CLaRK can process regular expressions bidirectional, either left-to-right or right-to-left, such that the direction of processing should be set firstly.

of POS tags, relatively productive grammar rules with partially lexicalized sequences, and less productive grammar rules with fully lexicalized sequences to recognize frozen strings. To make the grammar rules more general and productive, wild card symbols can be used in the regular expressions: such as @ for zero or one symbols, % for zero or more symbols, and # for one or more symbols.

The first 50 sentences of our data set are shallow processed with the regular expressions defined for initialization. After automatic annotation of this set of sentences, their annotations are finished manually to have trees like Figures 1 and 2. The result is a very small treebank.

In the next step, we extract all the grammar rules from this set of annotated sentences (50 sentences). The extracted rules are distinguished from this respect whether they are already defined in the system, or the annotator applied manually. To make this distinction explicit, the attribute '*r*' is added to the grammar rules defined as regular expressions in CLaRK. Absence of attribute '*r*' means that the rule is applied manually. Now we have two sets of rules: (1) a set of grammar rules which was already defined in CLaRK as regular expressions and it is not required to define it again; and (2) another set of grammar rules are added in the manual annotation step and they should be defined as new regular expressions in the system to further annotate new sentences. To select the grammar rules from the second set, this question may raise: "Which of the extracted grammar rules has a priority over the other rules to be defined as regular expressions in the CLaRK system?"

In cascaded regular grammar, the product of one rule is the input into another rule and in each rule only a portion and not the whole string is recognized. The property of the cascaded regular grammar is that there is an ordering on the hierarchy of the grammar rules; and the next rule is not applied unless the local conditions are satisfied. Thus, after defining the extracted grammar rules manually as new grammar rules in CLaRK, they are ordered hierarchically. Since the parsing strategy in CLaRK is bottom-up, the grammar rule extraction approach and ordering are also bottom-up; i.e. the grammar rules belonging to the higher level of the hierarchy are extracted and defined in CLaRK in a condition that the grammar rules belonging to the lower level of the hierarchy are already extracted and defined. As a result, the grammar rules in the lower level of the hierarchy have priority over the grammar rules in the higher level of the hierarchy. After finding the grammar rules which belong to the lower level of the hierarchy, a second filtering should be done in order to find the most frequent grammar rules frequently used by a human annotator and to assign a priority to them based on their frequency.

Figure 3 illustrates this idea on example (1). After extracting the grammar rules from the tree structures, they are ordered hierarchically. As can be seen in the figure, the extracted grammar rules from the tree structures are organized into several hierarchical levels in the bottom-up strategy. In the first step, the grammar rules of the lower level of the hierarchy are extracted. In the figure, the grammar rules of this level are shown with a solid-line triangle. The grammar rules of this level are the ones added to the system in the initialization step to handle multi-tokens and frequent phrasal constituents. After completion of extracting the grammar rules at this level, the grammar rules of the next level in the hierarchy are extracted. The result of the grammar rule extraction process is that the rules are classified into several hierarchical levels with respect to their seen constructions. The process of extracting the grammar rules continues iteratively until it reaches to the highest level of the hierarchy which is labeled S where it halts.

Since not all grammar rules are equally effective, we do not treat the grammar rules of each level equally. As a result, the grammar rules which are very productive and have high frequency of usage should be defined in the system. Based on this idea, the extracted grammar rules are sorted in a descending order of their frequencies, and the most frequent rules which have satisfied all the prerequisites are defined in CLaRK. The prerequisites are in fact the grammar rules in the lower levels of the hierarchy which are already extracted.

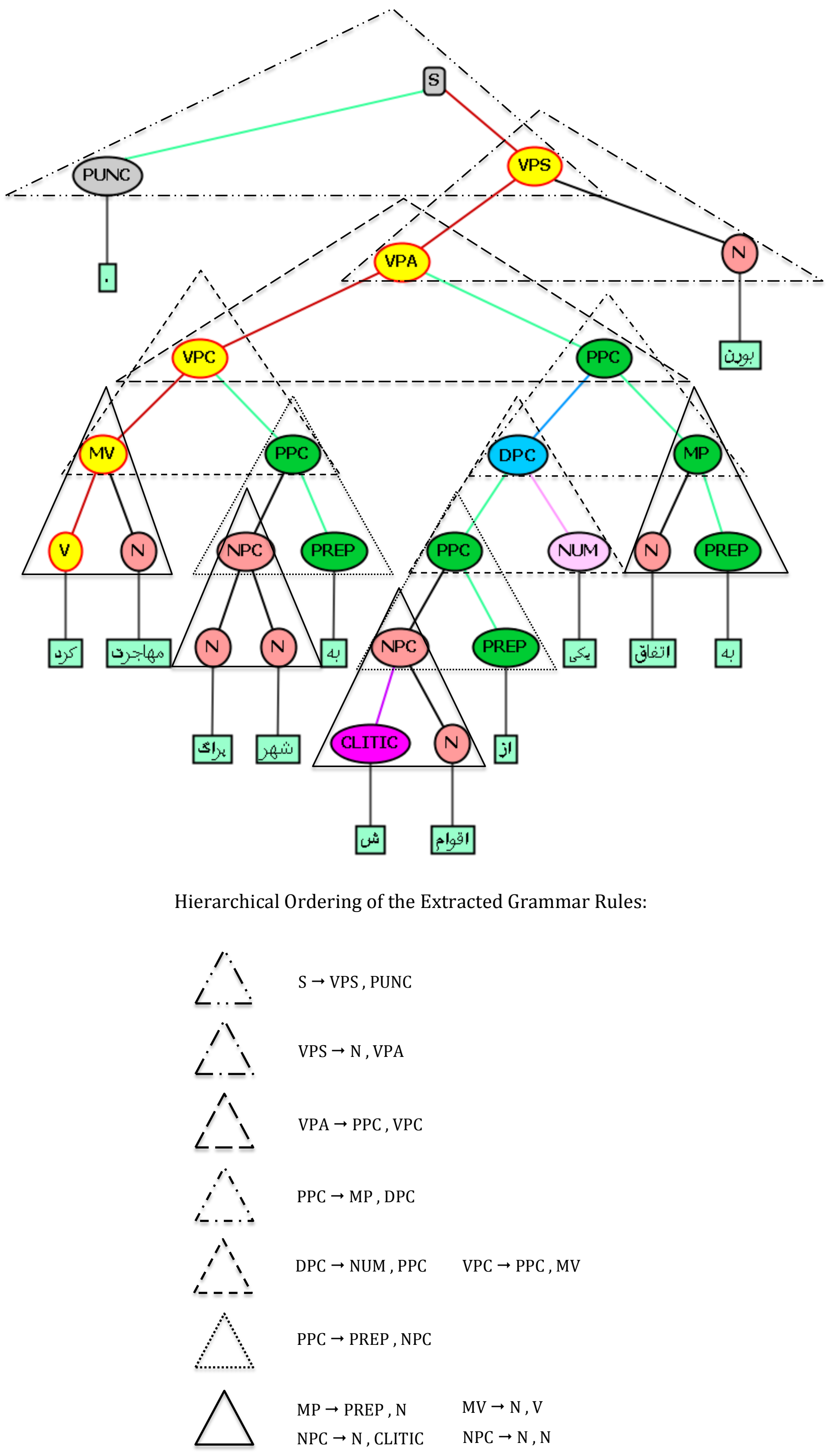
5. Bootstrapping Process for Treebanking

Algorithm 1 is used to bootstrap defining the grammar rules in CLaRK for further data annotation.

Algorithm 1 Bootstrapping Process in Treebank Development

```
Input: Set of Sentences from the Bijankhan Corpus,
       Set of Seed Grammar Rules $R_s$ defined in CLaRK
while all sentences are added to the treebank do
    Choose $N$ sentences $S$ from the corpus
    Use $R_s$ to annotate $S$ automatically
    Complete the annotation of $S$ manually
    Add the annotated $S$ to Treebank $T$
    Extract all applied manual grammar rules $R_m$ from $T$
    Select the $K$ most frequent grammar rules from $R_m$
    Define the $K$ grammar rules as regular expressions in
    CLaRK
    Augment $R_s$ with the $K$ grammar rules and remove them
    from $R_m$
end while
```

In this approach, firstly the set of 1000 sentences are preprocessed to realize multi tokens. Then, the seed grammar rules (R_s), the phrasal constituent grammar rules constructed from the bigram information, are defined as regular expressions in the system to start the annotation process. Applying R_s to 50 sentences provides a shallow processing of this set of data. The production of R_s is checked not to over-generate. In case of over-generation, constraints are imposed on it to limit its domain to the local context. Then, the annotation of these 50 sentences are completed manually. The bootstrapping process is started after defining the seed grammar rules in the system. To this aim, the remained sentences (950 sentences) are segmented into sets containing N sentences (*N=10*). In each iteration of the process, the set of 10 sentences is annotated with the seed grammar rules. The next step is the manual annotation to have a complete analyses of the sentences. In this step, based on what is described in Section 4, we extract the grammar rules applied manually (R_m) from the annotated sentences; then they are classified into several hierarchical levels, and in each level they are sorted in a descending order of their fre-



Hierarchical Ordering of the Extracted Grammar Rules:

S → VPS , PUNC

VPS → N , VPA

VPA → PPC , VPC

PPC → MP , DPC

DPC → NUM , PPC   VPC → PPC , MV

PPC → PREP , NPC

MP → PREP , N   MV → N , V
NPC → N , CLITIC   NPC → N , N

Figure 3: Extracting the grammar rules and ordering them wrt to their hierarchical levels

quency. Then, the K most frequent grammar rules (K=5) of one level which have satisfied all the prerequisites are defined as new regular expressions in CLaRK. R_s is augmented with the newly selected grammar rules from R_m. Finally, the modified version of R_s is used for the annotation of the next iteration. After extracting the grammar rules in each iteration, they are checked whether they are already defined in the system or not. If yes, they are removed from the list of extracted grammar rules to increase the chance of the grammar rules which are not defined in the system yet. The bootstrapping process continues iteratively, and it terminates when the whole set of selected sentences (1000 sentences) are annotated completely. The result is a treebank which is developed with a bootstrapping process via defining new grammar rules extracted from a small set of annotated data. It needs to be added that after applying the grammar rules automatically and before starting the manual annotation, a rule might over-generate; and since in HPSG the argument structure of a lexicon plays the most important role to define the type of the dependents, the parse results should be corrected manually.

6. Experimental Results

Using the model described in Algorithm 1, we built a treebank for Persian composed of 1000 sentences. In average, the length of each sentence in the treebank is 27.67 words. To annotate each of these sentence completely, 28.03 grammar rules are required generally, out of which 12.19 grammar rules are applied automatically and the rest manually. Table 4 summarizes the information about the developed treebank.

Table 4: Summary of the information about the developed treebank for Persian

num. of sentences	average length of sentences (words)	average num. of automatic rules	average num. of manual rules
1000	27.67	12.19	15.84

To evaluate the overall performance of the grammar rules used in our method for data annotation, precision and recall are computed. To compute precision, the correct grammar rules (GR) which are applied automatically against the total number of grammar rules applied automatically are measured:

$$precision = \frac{\#\ of\ correct\ automatic\ GR}{total\ number\ of\ automatic\ GR}$$

To compute recall, the correct grammar rules which are applied automatically are measured against the sum of correct automatic grammar rules and the manual grammar rules:

$$recall = \frac{\#\ of\ correct\ automatic\ GR}{\#\ of\ correct\ automatic\ GR\ +\ manual\ GR}$$

f-score is used to weight the average of precision and recall:

$$f_1 - score = \frac{2\ *\ precision\ *\ recall}{precision\ +\ recall}$$

The summary of the result is reported in Table 5. As can be seen from the results, 86.20% of the automatic grammar rules are correctly applied which has the coverage of 45.65% over all the correct automatic grammar rules used to annotate the sentences.
Based on the results, precision is relatively high; but recall is low because of two reasons: (1) there are contexts that are not seen so far, and the relevant grammar rules are not defined in the system as a result; (2) even if a grammar rule is defined, there is no grantee that it is added to the system via our bootstrapping process because of its low frequency. Of course annotating more sentences is a help to increase recall, since having more annotated data and more grammar rule definition increase the coverage of the total rules.

Table 5: Evaluation results

Precision	Recall	f-score
86.20	45.65	59.69

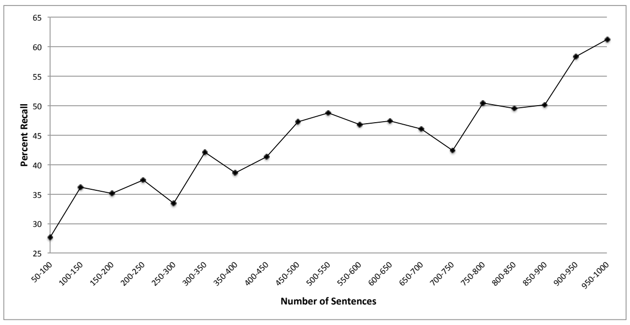


Figure 4: Recall measured every 5 iterations

To verify this statement, we measured recall of our method for every 5 iterations and presented it in Figure 4. As can be seen in this figure, recall is constantly increasing which is a signal that newly defined grammar rules in each iteration have a positive effect on the coverage, and even higher recall can be achieved in further iterations. There are two considerable drops in the graph. Consulting the data, we found that extraposition, scrambling, and ellipses frequently happened in these portions of data, and a higher amount of human intervention was required to complete the analyses. Moreover, errors in the assigned POS tags had a negative effect on applying the grammar rules. Additionally, changing the genre and the domain of the texts from politics to economy or sport news were other factors that affected recall.

7. Summary

This paper described a grammar rule extraction task which helped to develop a treebank via a bootstrapping process. The introduced method is very useful and practical for languages like Persian that suffer from lack of available data sources including treebanks, and the data annotation task has to be started from scratch. The advantage of this method is a constant increase on the automatic annotation which results in the reduction of human intervention during the data annotation process.

8. Acknowledgements

Masood Ghayoomi is funded by the German research council DFG under the contract number MU 2822/3-1.

9. References

Steven Abney. 1996. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337–344.

Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. 2011. Lessons from building a Persian written corpus: Peykare. *Language Resources and Evaluation*, 45(2):143–164.

Mahmood Bijankhan. 2004. The role of corpora in writing grammar. *Journal of Linguistics*, 19(2):48–67. Tehran: Iran University Press.

Paul J. King and Kiril Simov. 1998. The automatic deduction of classificatory systems from linguistic theories. *Grammars*, 1(2):103–153.

Geoffrey Leech and Andrew Wilson, 1999. *Standards for Tagsets*, pages 55–80. Text, speech, and language technology. Kluwer Academic Publishers, 9 edition.

Mahdi Mohseni and Behrouz Minaei-bidgoli. 2010. A Persian part-of-speech tagger based on morphological analysis. In *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC'10)*, pages 1253–1257, Valletta, Malta.

Mahdi Mohseni. 2008. Automatic part-of-speech tagging and disambiguation system for the textual corpus of the persian language. Master's thesis, University of Science and Technology: Department of Computer Science, Iran.

Stefan Müller and Masood Ghayoomi. 2010. PerGram: A TRALE implementation of an HPSG fragment of Persian. In *Proceedings of 2010 IEEE International Multiconference on Computer Science and Information Technology*, pages 461–467, Wisła, Poland.

Stefan Müller, Pollet Samvelian, and Olivier Bonami. In Preparation. *Persian in Head-Driven Phrase Structure Grammar*.

Stefan Müller. 2010. Persian complex predicates and the limits of inheritance-based analyses. *Journal of Linguistics*, 46(3):601–655.

Petya Osenova and Kiril Simov. 2003. The Bulgarian HPSG treebank: Specialization of the annotation scheme. In *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*.

Gerald Penn. 2004. Balancing clarity and efficiency in typed feature logic through delaying. In *Proceedings of ACL 2004*, pages 239–246.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Pollet Samvelian and Jesse Tseng. 2010. Persian object clitics and the syntax-morphology interface. In Stefan Müller, editor, *Proceedings of the 17th International Conference on Head-Driven Phrase Structure Grammar*, pages 212–232. CSLI Publications.

Pollet Samvelian. 2007. A (phrasal) affix analysis of the Persian Ezafe. *Journal of Linguistics*, 43:605–645.

Kiril Simov, Zdravko Peev, Milen Kouylekov, Alexander Simov, Marin Dimitrov, and Atanas Kiryakov. 2001. CLaRK - An XML-based system for corpora development. In *Corpus Linguistics Conference*, pages 558–560, Lancaster,UK.

Kiril Simov, Alexander Simov, Milen Kouylekov, Krasimira Ivanova, Ilko Grigorov, and Hristo Ganev. 2003. Development of corpora within the CLaRK system: The BulTreeBank project experience. In *Proceedings of the Demo Sessions of the 10th Conference of EACL*.

Kiril Simov. 2001. Grammar extraction from an HPSG corpus. In *Proceedings of the RANLP 2001 Conference*, pages 285–287, Tzigov Chark, Bulgaria.

Kiril Simov. 2002. Grammar extraction and refinement from an HPSG corpus. In *of the ESSLLI Workshop on Machine Learning Approaches in Computational Linguistics*, pages 38–55, Trento, Italy.

Kiril Simov. 2003. HPSG-based annotation scheme for corpora development and parsing evaluation. In *Proceedings of the RANLP 2003 Conference*, pages 432–439, Borovets, Bulgaria.

Mehran A Taghvaipour. 2005. *Persian Relative Clauses in Head-driven Phrase Structure Grammar*. Ph.D. thesis, Department of Language and Linguistics, University of Essex.

Appendix 1

- Ordered Information of POS tags for **Clitics**: category(Clitic(C)); type(enclitic(e)/ proclitic(p)); minor POS(pronoun(z)/ verb(v)/ post-position(p)/ conjunctor(j)/ preposition(e)/ adverb(d)/ determiner(t)); polarity(positive(p)/ negative(n)); verb-type(copula(k)); host (adjective(a)/ noun(n)/preposition(e)/ adverb(d)/ pronoun(z)/ number(u)/ conjunctor(j)/ past-participle(p)/ abbreviation(b)/ residual(s)/ punctuation(o)/ determiner(t)/ cimperfect(q)/ imperfect(i)/ classifier(l)/ interjection(r)/ clitic(c)/ perfect(f)/ post-position(m)/ verb(v)); number(singular(s)/ plural(l)); person(first(o)/ second(t)/ third(h)); final letter of the host(consonant(c)/ vowel(v))

- Ordered Information of POS tags for **Nouns**: category(Noun(N)); type(common(c)/ proper(a)); number(singular(s)/ plural(l)); polarity(negative(n)/ positive(p)); semantic(location(k)/ time(t)/ day(d)/ month(m)/ season(e)/ title(f)/ vocalic(v)/ direction(c)); abbreviation(b); clitics(Ezafe(z)/ indefininite(y))

- Ordered Information of POS tags for **Numbers**: category(Number (U)); type(ordinal(o)/cardinal(r)); local function(adjective(a)/ noun(n)); number(singular(s)/ plural(l)); fusion(adjective(a)/ classifier(l)); clitics(Ezafe(z)/ indefininite(y)); semantic(time(t))

- Ordered Information of POS tags for **Prepositions**: category(Preposition(E)); clitics(Ezafe(z)/ indefininite(y))

- Ordered Information of POS tags for **Verbs**: category(Verb (V)); polarity(positive(p)/ negative(n)); auxiliary or main verb(auxiliary(x)/ main(y)); type(simple(s)/ copula(k)/ infinitive(i)/ light(b)); number(singular(s) /plural(l)); person(first(o)/ second(t)/ third(h)); tense(present(s)/ past(t)/ future(u)); aspect(perfect(f)/ imperfect(i)/ cimperfect(q)); mood(subjunctive(u)/ imparative(m)/ (past-participle(p)); clitics(Ezafe(z)/ indefininite(y)); impersonal modal(n)