From HPSG-based Persian Treebanking to Parsing: Machine Learning for Data Annotation

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Degree of
Doctor of Natural Sciences (Dr. rer. nat.)
to the Department of Mathematics and Computer Science
of Freie Universität Berlin

by

Masood Ghayoomi

Berlin, 2014

Supervisor:	Professor Dr. Raúl Rojas
Second examiner:	Professor Dr. Stefan Müller
Third examiner:	Professor Dr. Kiril Ivanov Simov
Date of the defense:	June 27, 2014

Affidavit

I hereby swear in lieu of an oath that I have independently prepared this dissertation and without using other aids than those stated. The data and concepts taken over from other sources or taken over indirectly are indicated with citations. This dissertation so far has not been submitted in either Germany or any other country in the same or a similar form in a procedure for obtaining an academic title.

Masood Ghayoomi
Berlin, 2014

To my wife Saeedeh,

who is my love.

To my daughter Noora,

who is my life.

Acknowledgements

No worthwhile work has ever completed in isolation. Finishing this dissertation could not have been possible without the help of several mentors, colleagues, friends, and relatives. I owe them a debt of gratitude for all of their support and kindness.

I am very grateful to Professor Dr. Raúl Rojas, my first reviewer, Professor Dr. Stefan Müller, my second reviewer, and Professor Dr. Kiril Ivanov Simov, my third reviewer, for their supervision and reviewing my dissertation. I am truly thankful for their patience and their encouragement. They taught me a tremendous amount about how to do good research. Their review of my dissertation greatly improved my work to develop a clearer insight and explanation of what I have done in my research. I would like to thank Professor Dr. Marco Block-Berlitz, PD Dr. Christoph Benzmüller and Dr. Tim Landgraf for serving on my committee.

I am also grateful to Professor Dr. Jonas Kuhn for acting as my supervisor when I was visiting the Institute for Natural Language Processing (IMS) at the University of Stuttgart. His invaluable suggestions had a major impact on my work.

I would like to thank Professor Dr. Kiril Ivanov Simov and Professor Dr. Petya Osenova for their advice during my visit at the Linguistics Modeling Department at the Bulgarian Academy of Sciences, Sofia, Bulgaria. In addition to being my academic mentors, they have also been great friends to me.

My special appreciation goes to Professor Dr. Mahmood Binajkhan (University of Tehran, Tehran) and Professor Dr. Vida Shaghaghi (Allameh Tabatabaei University, Tehran), and also to Professor Dr. Omid Tabibzadeh (Bu-Ali-Sina University, Hamedan) for their guidance and the interesting discussions that we had on Persian and the complex data analysis.

I greatly appreciate my colleagues in the German Grammar Group at the Freie Universität Berlin (FU-Berlin) for their true friendship and the fruitful conversations we had during my work: Dr. Felix Bildhauer, Dr. Bjarne Ørsnes, Dr. Philippa Cook, Dr. Roland Schaefer, Dr. Jakob Maché, and Janna Lipenkova.

My thanks also go to my colleagues in IMS at the University of Stuttgart for the interesting discussions that we had: Dr. Helmut Schmid, Dr. Richard Farkas, Wolfgang Seeker, and Anders Björkelund.

I would also like to thank Dr. Ines Rehbein (Potsdam University), Dr. Yi Zhang (Nuance Communications Aachen GmbH), Dr. Mehran Taghvaipour, and Thomas Triebwasser.Prado for the discussions, comments, and proofreading the draft version of the dissertation. All remaining errors are of course mine. I also appreciate Omid Moradiannasab to help me in programming.

I am very thankful to the German organizer Deutsche Forschungsgemeinschaft (DFG) who provided me with the financial support for this research. During my study, I was funded by DFG as a research assistant for three years (2009-2012) in the project entitled "Implementation of a Head-driven Phrase Structure Grammar of Persian and Creation of a Machine Readable Lexicon of Persian" under the contract number MU 2822/3-1 at FU-Berlin. I was also funded by DFG in the Integrated Graduate School for one year (2012-2013) within Sonderforschungsbereich 732 (SFB 732) on "D2: Combining Contextual Information Sources for Disambiguation in Parsing and Choice in Generation" at the University of Stuttgart.

In addition to the people who provided me with scientific support, I would like to acknowledge the people who provided me a conducive environment and the condition to do my research: Viola Auermann, the Secretary of the German Grammar Group at FU-Berlin, Dr. Jagoda Bruni, the Coordinator of the Integrated Graduate School at IMS, and the ZEDAT, the Computer Center at FU-Berlin, for their help and support with the server facilities.

This dissertation is written in the XƎLaTeX environment. I would like to thank all package providers who facilitated compiling my dissertation very nicely.

In my educational life, my beloved family has always been a great support both spiritually and mentally. During the time that I was away from them, they did all the best they could to keep me motivated to pursue my education. My special gratitude goes to my parents, Naser Ghayoomi and Esmat Mehrabi Ghomi, who always encouraged me to pursue my education. I also appreciate my parents-in-law for their encouragement to seek knowledge. Without their support, I could not have been where I am now. No words can express my gratitude to my family for what they have done for me throughout my life. Last, but not least, my appreciation goes out to my beloved wife, Saeedeh Momtazi, for her true love and emotional support. She took care of me in my Ph.D. education beside being a mother for our new born daughter, Noora. Being a graduate student limited the time that I could spend with them as a husband and a father. I am grateful to my wife for all her patience, tolerance, and understanding. Her ceaseless positivity has always been the best motivation for me to keep working.

Short Summary

Parsing is a step for understanding a natural language to find out about the words and their grammatical relations in a sentence. Statistical parsers require a set of annotated data, called a treebank, to learn the grammar of a language and apply the learnt model on new, unseen data. This set of annotated data is not available for all languages, and its development is very time-consuming, tedious, and expensive. In this dissertation, we propose a method for treebanking from scratch using machine learning methods.

We first propose a bootstrapping approach to initialize the data annotation process. We aim at reducing human intervention to annotate the data. After developing a small data set, we use this data to train a statistical parser. This small data set suffers from the sparseness of data at the lexical and syntactic construction levels. Therefore, a parser trained with this amount of data might have a low performance in a real application. To resolve the data sparsity problem at the lexical level, we propose an unsupervised word clustering approach to provide a more coarse-grained representation of the lexical items. To resolve the data sparsity problem at the syntactic construction level, we propose active learning which is a promising supervised method to seek informative samples in a data pool. The data that is annotated through an active learning approach helps a learner to obtain performance similar to that of a learner trained with the complete set of annotated data. Consequently, active learning is a great help to reduce the amount of required annotated data.

Kurze Zusammenfassung

Parsing bezeichnet einen Schritt in der automatischen Analyse natürlicher Sprache, bei dem die grammatischen Relationen zwischen den Wörtern eines Satzes offengelegt wird. Um die Grammatik einer Sprache zu lernen und auf neue Daten anwenden zu können, werden statistische Parser auf syntaktisch annotierten Daten, sogenannten Baumbanken, trainiert. Baumbanken sind nur für wenige Sprachen verfügbar, denn die manuelle Erstellung von Baumbanken ist langwierig, mühsam und teuer. Diese Dissertation präsentiert eine Methode zur Erstellung von Baumbanken mit Hilfe maschinellen Lernens.

Als erstes schlagen wir einen Bootstrapping-Ansatz vor, welcher den Anteil an menschlicher Intervention zu minimieren sucht. Dabei verwenden wir eine kleine Menge eigens dafür annotierter Daten, um einen statistischen Parser zu trainieren. Bedingt durch die geringe Größe des Datensatzes zeigt das so trainierte Modell Abdeckungsprobleme auf der lexikalischen und syntaktischen Annotationsebene und würde in einer realistischen Anwendung schlechte Qualität liefern. Um die lexikalische Abdeckung zu verbessern, schlagen wir deshalb einen unüberwachten Wort-Clustering-Ansatz vor, welcher Wörter im Lexikon zu größeren Klassen gruppiert. Um die syntaktische Abdeckung zu verbessern, verwenden wir Active Learning, eine überwachte Methode um informative Beispiele aus einem großen Pool unannotierter Daten auszuwählen. Die Verwendung von Active Learning bei der Datenannotation reduziert die Menge an Daten, die annotiert werden muß, bevor man dieselbe Performanz erreicht, die man unter Verwendung der Gesamtdatenmenge erreichen würde. Active Learning ist deshalb eine große Hilfe, wenn man die Menge an zu annotierenden Daten reduzieren will.

Summary

Natural language understanding is a subfield of computational linguistics which aims at making machines able to understand human languages. Obtaining knowledge of the grammar of the language in question through parsing tasks plays an important role in achieving this goal. A treebank is required to train a statistical parser to induce and learn the grammar of a language. This set of annotated data, however, does not exist for all languages. The main motivation behind the present research is to propose a method to develop a treebank from scratch for Persian. To this end, we use machine learning methods to reduce human intervention to annotate data in the framework of the Head-driven Phrase Structure Grammar (HPSG). This process is done in two phases. In the first phase, a small treebank is developed to train a statistical parser, and in the second phase, the problems one might face in employing the trained parser in a real application are taken into consideration and solutions are proposed to resolve the problems.

To initialize the data annotation process, we use a bootstrapping approach to develop a small treebank. In this method, a set of seed data is developed with minimum human effort for initialization. Next, all grammar rules are extracted from the annotated data and the most frequent ones are defined as regular expressions in an annotation tool to be used for further data annotation. This approach continues iteratively until the desired size of data is annotated. An HPSG-based annotation scheme is exploited during the annotation process.

After developing this set of annotated data, it can be used for training statistical parsers. When using the trained parser in a real application, it suffers from the sparseness of data at two levels, namely the lexical and syntactic construction levels. To reduce the data sparsity problem at the lexical level, we propose a class-based model. In this model, we use a raw corpus and the Brown word clustering algorithm (Brown et al., 1992) to cluster the words. Then, we map the words in the treebank to the corresponding clusters, and retrain the parser. One drawback of the Brown word clustering algorithm is treating homographs the same. Since in Persian short vowels are not written, there are a large number of homographs. We extend the Brown word clustering algorithm by attaching the main part-of-speech tag of the words to the word forms before clustering. In this data format, homographs are treated relatively distinctively. The experimental results show that the extended clustering model outperforms the normal one.

To reduce the data sparsity problem at the syntactic construction level, we use active learning, which is a supervised machine learning method. In this model, we seek informative samples in a data pool and ask a human for their annotation. One metric for selecting informative samples is computing the entropy of tree analyses provided by the parse, and the ones with high entropy are selected as informative samples. Another metric is using a committee of learners. If parse trees of a sentence, which are the output of two or more parsers, have a minimum agreement among the committee members, that sentence can be selected as an informative sample. Yet another criterion is computing the rate of tree similarity between the tree analyses of sentences in the data pool and the training data for selecting the informative samples. In this metric, the most dissimilar sentences are the candidates to be selected through the active learning process. All of the proposed active learning models are tested and the results are compared. Our experimental results show that among all of the proposed models, the word-based tree similarity model and the query-by-committee sampling model, which is composed of word- and class-based parsing, obtain a relatively better performance with the minimum number of lexical items to be learnt by the parser.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Natural language understanding is one of the active subfields in computational linguistics. It seeks to enable computers to learn and understand human languages. Understanding the grammar of a language is one of the key steps towards understanding a natural language. This goal is achieved by passing through the syntactic analysis channel, because in this phase of the analysis, the syntactic categories of words and the interaction of the words to construct a sentence are determined. Furthermore, it is a fundamental step towards semantic and discourse processing. This channel has two components: (a) part-of-speech (POS) tagging for identifying the syntactic category of each word in the local context, and (b) parsing for providing the syntactic analyses of sentences and identifying the relations between the words that construct sentences. Both of these components can be developed through the rule-based, the statistical, or the hybrid approach.

One of the advantages of the rule-based approach is that no annotated data is required. This approach, however, requires grammar rules written by a grammarian. All of the written grammar rules might not be reusable for other languages due to differences between languages. Another disadvantage of this approach is that if a grammar rule does not exist for analyzing a word or a sentence, then the system cannot deal with it, and it halts.
The statistical approach can be used either without human supervision (unsupervised method) or with human supervision (supervised method) to process the language automatically. The main advantage of the statistical approach over the rule-based approach is that it can automatically adapt to new domains, genres, or languages for inducing the linguistic properties and building a grammar model. In the unsupervised method, no preliminary linguistic knowledge

is required, and the system induces grammar rules automatically to create the grammar model. Systems developed based on this approach are not expensive to develop, but they usually do not perform well. With the supervised method, annotated data is required to induce the grammar rules from the data and create a grammar model to analyze unseen data. Developing the required annotated data for the supervised method is time-consuming and expensive since it requires human effort. The drawback of this approach is its high degree of sensitivity to the quality and quantity of the annotated data. The advantage of this method is that it has a higher performance compared to the unsupervised method, because correctly annotated data is used for creating the grammar model. The advantages of the supervised method motivated us to use it in our study for processing the Persian language.

1.2 Problems

As described above, the development of annotated data and its availability are crucial for linguistic investigations and also data driven approaches in human language technologies. Certain languages such as English and German have been given a great deal of attention. This has resulted in the availability of many types of data sources for these languages. The developed data sources can be used for training tools in different applications. However, other languages like Persian are less developed in terms of the availability and accessibility of such annotated data. Developing data sources is one issue, and their public availability and free accessibility for their reusability are other issues that need to be taken into consideration.

In this dissertation, we aim at bridging the gap by proposing a method to develop a treebank for Persian from scratch and to make the developed treebank freely available online. This data source can be used as an annotated corpus to train statistical parsers. Since we want to use a supervised approach in our study for processing Persian, human effort is required to develop the treebank, which is a very labor-intensive, time-consuming, tedious, and expensive task. Due to the hardness of the data annotation task, we develop a small treebank in the first step. This data set is then used for training a statistical parser. When using the parser in a real application to parse unseen data, low performance might result. The reason the parser might not perform well is due to the data sparsity problem at the lexical and syntactic construction levels.

A treebank is an annotated corpus which consists of a set of sentences with their corresponding syntactic analysis. This set of annotated data is errorless, and it contains only grammatical sentences or phrases. This data set should be the representative of the target language, in terms of both the lexicon and the syntactic constructions that exist in the language. Taking the properties of such a corpus into consideration, a treebank, even the most expansive one, does not cover all properties of a language and it cannot be a good representative of a language because it suffers from the sparseness of data. While developing a treebank, we address the data sparsity problem, and we try to provide answers for the following questions throughout our study.

- Considering that human effort is required for annotating data, is there any way to reduce the labor-intensity of data annotation and reduce the amount of human intervention in the development of annotated data?
- Considering that a statistical parser trained with a treebank suffers from the sparseness of data, is there a way to reduce the sparseness of the lexical items and syntactic constructions in a treebank? A simple answer to this question might be increasing the size of the annotated data. But it may in turn raise questions over how we can control the properties of sentences to be added to the treebank. Should any sentence be added to the treebank or should we be selective and seek sentences which are informative?
- A treebank contains sentences with their corresponding linguistic analysis, and a more accurate grammar model is built with larger amount of training data. This question may raise whether there exist other factors except the size of the training data that affect the parsing performance. How fine- or coarse-grained should the linguistic information be in a treebank? What is the impact of the treebank annotation granularity on parsing?

1.3 Contributions

As mentioned, the development of a data source is very labor-intensive, time-consuming, tedious, and expensive. To reduce the amount of human effort, we propose a method to use machine learning techniques for reducing the degree of human intervention in data annotation and increasing the speed of the annotation process consequently. To this end, a bootstrapping approach is proposed to start the data annotation process from scratch and incrementally develop a treebank for Persian. In this approach, the most frequent grammar rules of a Persian corpus are selected from the pool of extracted grammar rules to be automatically applied at unannotated data and reduce human effort for the annotation task. The developed treebank can be used for training a statistical parser.[1] To this end, we adapt different statistical parsers to be trained by this

[1]It should also be noted that our developed data set is also used for training a statistical parser developed by Sarabi and Analouie (2012).

data set. The treebank is developed based on the Head-driven Phrase Structure Grammar (HPSG) formalism (Pollard and Sag, 1994).

As noted, using this parser in a real application suffers from the data sparsity problem at both the lexical and syntactic construction levels. To handle the sparseness of lexical items, we propose using an unsupervised clustering approach to cluster the words into classes and to use this class-based model for parsing rather than the original words in the treebank. In this approach, the words with similar features are assigned to the same cluster, and it helps to reduce the lexicon size. To deal with the sparseness of the syntactic constructions, we propose using active learning.
Active learning is a supervised machine learning approach to find the difficult samples from a data pool in which the syntactic constructions of the selected examples do not exist in the grammar model, and it asks an oracle to annotate the selected data. In the sample selection step, we aim at selecting the informative samples add them to the treebank. Since not all sentences are interesting to be annotated in the treebank due to the existing redundancy in syntactic constructions, sampling has to be done more deliberately. Therefore, the sentences in which their syntactic constructions do not exist in the treebank should be selected. This approach results in minimization of the treebank size while still capturing the most informative samples from rare to frequent syntactic constructions in the corpus.

In addition to solving the above problems, we study the effect of the treebank annotation granularity (fine- vs coarse-grained annotation) on parsing in three dimensions: the lexical item, the POS tag, and the constituent label. The main contribution of this is studying the effect of different variables on parsing performance besides the data sparsity problem.

Another contribution of this research is introducing an algorithm for automatically converting our developed HPSG-based treebank into its parallel dependency-based treebank. We further evaluate the performance of the dependency parsers trained with the converted treebank.

The preliminary study of this research was started in October 2009, when there was no treebank available for Persian. We selected the HPSG formalism (Pollard and Sag, 1994) as the backbone of the treebank to develop a data source that contains both the constituents' hierarchies along with head-daughter dependency relations. This treebank was released online[2] in October 2011, and its availability and accessibility were publicly announced on January 22, 2012.[3] A short-while after this announcement, the first dependency treebank of Persian

[2]`http://hpsg.fu-berlin.de/~ghayoomi/PTB.html`

[3]`http://linguistlist.org/issues/23/23-395.html` (Accessed:19/08/2014)

developed by the Dādegān[4] group was also released for free.[5]

1.4 Dissertation Overview

This dissertation is composed of nine chapters in two parts. The structure of the dissertation is as follows.

Part I mainly focuses on the linguistic perspective of our study by providing linguistic theories and background.

Chapter 2 basically describes the properties of the Persian language, its syntax, and the available language resources. Next, the problems of processing a Persian text that should be resolved in the pre-processing step are categorized and explained in detail.

Chapter 3 reviews the literature and provides background. The content of this chapter covers the basic concepts in three dimensions. One dimension is describing briefly some of the well-known grammar formalisms that have applications in computational linguistics. The HPSG formalism and the previous studies on Persian HPSG are also discussed. In another dimension, treebanking is defined and the previous studies on major treebanks in general and the Persian treebanks in particular are described and discussed. In the third dimension, parsing and its methods, the evaluation metrics, and the previous studies on parsing Persian sentences are discussed.

Chapter 4 covers the development of an annotation scheme to be used throughout our treebanking process. Additionally, the order of realizing the dependency relations and the element sets used in the treebank are defined. Furthermore, to make the description of the annotation scheme precise, samples with their corresponding tree analyses for the target syntactic constructions are provided.

Part II mainly focuses on the computational perspective of our study by providing computational approaches for processing the Persian language.

Chapter 5 describes the bootstrapping approach used for treebanking. In this chapter, the tool and the bootstrapping algorithm employed for the treebank development are explained. In the bootstrapping process, the most frequent, definite grammar rules are extracted from the treebank and defined as regular expressions in the annotation tool to further annotate unseen data. The annotation process is done in four steps: (1) pre-processing: to make the data

[4]`http://dadegan.ir`

[5]`http://mailman.uib.no/public/corpora/2012-January/014823.html` (Accessed:19/08/2014)

ready, (2) initialization: to provide seed data, (3) main-processing: to shallowly annotate the data automatically, and (4) post-processing: to complete the annotation of sentences manually. The amount of human intervention to annotate the data and the accuracy of the defined grammar rules in the tool are also evaluated.

Chapter 6 introduces the required tools, data sets, and data preparation for our study. To this end, the basic properties of Natural Language Processing (NLP) tools used for the experiments in our study are explained, such as the TnT and Stanford POS taggers, the Berkeley and Stanford constituency parsers, and the Malt and Mate dependency parsers, which are trained with our developed treebank and adapted to Persian. The tools can potentially be trained with three dimensions of annotation granularity, namely the lexical item, the POS tag, and the constituent label. In the rest of this chapter, the parsers' performance are evaluated and the obtained results are reported. Since the parsers are trained with a small amount of data, they suffer from the data sparsity problem at two levels, namely the lexical items and the syntactic constructions. Reducing the two sparsity problems are the topics discussed in detail in Chapters 7 and 8.

Chapter 7 proposes class-based models to reduce the data sparsity problem at the lexical level. To this end, the Brown word clustering algorithm (Brown et al., 1992) is used for clustering words in an unsupervised mode. We further propose an extension to this model to deal with homographs. Next, the performance of the models is compared. Additionally, other aspects of the annotation granularity which affect parsing are further investigated.

Chapter 8 deals with the data sparsity problem at the syntactic construction level. This chapter employs active learning for data annotation. Among the learning scenarios, a pool-based active learning scenario is used for our task to select the most informative samples from the data pool. Considering our ultimate goal to reduce the data sparsity problem at the syntactic construction level, we propose several models to select the informative samples. Next, we simulate active learning with our current data to show the usability of our models in a real application for further data annotation.

Chapter 9 concludes our study and presents suggestions for future studies.

1.5 Published Papers

The significant parts of this dissertation have been published in the following journals, conferences, and workshops:

- JOURNALS
  - International Journal on Asian Language Processing,
    (Ghayoomi et al., 2010).
  - Language Resources and Evaluation,
    (Bijankhan et al., 2011).
  - Linguistic Issues in Language Technology,
    (Ghayoomi, 2012a).

- CONFERENCES and WORKSHOPS
  - NAACL–HLT 2010 Workshop on Active Learning for Natural Language Processing, Los Angeles, USA,
    (Ghayoomi, 2010).
  - International Multi-conference on Computer Science and Information Technology – Computational Linguistics Applications, Wisła, Poland,
    (Müller and Ghayoomi, 2010).
  - 4th International Conference on Iranian Linguistics, Uppsala, Sweden,
    (Ghayoomi and Müller, 2011).
  - 8th International Conference on Language Resources and Evaluation, Istanbul, Turkey,
    (Ghayoomi, 2012b).
  - 8th International Conference on Advances in Natural Language Processing, Kanazawa, Japan,
    (Ghayoomi, 2012c).
  - 11th International Workshop on Treebanks and Linguistic Theories, Lisbon, Portugal
    (Ghayoomi and Moradiannasab, 2012).
  - 8th Conference of Iranian Linguistics, Tehran, Iran,
    (Ghayoomi, 2013).
  - 12th International Workshop on Treebanks and Linguistic Theories, Sofia, Bulgaria,
    (Ghayoomi and Kuhn, 2013).
  - 9th International Conference on Language Resources and Evaluation, Reykjavik, Iceland,
    (Ghayoomi and Kuhn, 2014)

Part I

Linguistic Theory and Background

Chapter 2

The Persian Language

2.1 Introduction

Text corpora are the electronic data sources used for linguistic investigations or NLP applications. A corpus may contain texts in a single language (monolingual corpus) or in multiple languages (multilingual corpus). Corpora are the main data sources in corpus linguistics for studying the linguistic properties of language(s) expressed in real world texts. There are several resources in written texts that can be used for developing text corpora. Newswires and books are the best-known resources for this task. Nowadays, web pages are also widely used as rich resources to construct corpora, because the collection of texts in various genres written by different authors is one of the main criterion of a corpus to be representative of a language. The goal of collecting texts from such resources is to provide a reasonably accurate picture of the entire language. Reaching this goal, however, is not an easy task. One might face problems while building a corpus from such resources, since not all texts are written in a standard form. Therefore, before processing the corpus, the text should be normalized, and the problems should be solved during the pre-processing step. In the development of a Persian corpus, we face a number of problems due to particular characteristics of this language, which are detailed in this chapter by representing a comprehensive perspective of developing a corpus for Persian and discussing the major problem areas that must be resolved to normalize a Persian text before processing.

This chapter contains six sections. First, the basic properties of the Persian language and its syntax are discussed in Sections 2.2 and 2.3. Next, the available language resources for Persian are introduced in Section 2.4. In Section 2.5, the various problems that one may face when processing a Persian text are categorized and explained. The discussion of this section is mainly important for the pre-processing step in the annotation process. Finally, the chapter is summarized in Section 2.6.

The content of this chapter is mainly based on the following papers:

- Ghayoomi, Masood and Saeedeh Momtazi and Mahmood Bijankhan (2010) “A study of corpus development for Persian” In *International Journal on Asian Language Processing*, 20 (1): 17–33.
- Bijankhan, Mahmood and Javad Sheykhzadegan and Mohammad Bahrani and Masood Ghayoomi (2011) “Lessons from building a Persian written corpus: Peykare” In *Language Resources and Evaluation*, 45 (2): 143–164, Springer.
- Ghayoomi, Masood and Stefan Müller (2011) “Multi-token units and multi-unit tokens in developing an HPSG-based treebank for Persian” In *Proceedings of the 4th International Conference on Iranian Linguistics*, June 17–19, 2011; Uppsala University, Uppsala, Sweden, p: 29.

2.2 About the Language

Persian is a member of the Iranian branch of the Indo-European language family, and it has many features and properties in common with other members in terms of morphology, syntax, phonetics, and lexicon. Even though Persian adopted the Arabic script, it differs fundamentally from the Arabic language, which is in the Semitic language family. While Persian has 32 alphabetical letters, Arabic only has 28. Persian has four more letters than Arabic, namely PE (‘پ’), CHE (‘چ), ZHE (‘ژ’), and GĀF (‘گ’).[1] Like Arabic, Persian letters have joiner or non-joiner forms based on their positions in a word. Most of the letters have four forms. In three positions (the beginning, the middle, and the end of a word), the letters appear in joined form attached to the neighboring letter(s). The other form is an isolated non-joiner which appears at the very end of a word. For instance, the letter EYN is written ‘عـ’ for the beginning joiner, ‘ـعـ’ for the middle joiner, ‘ـع’ for the end joiner, and ‘ع’ for the non-joiner. Tables A.1 and A.2 in Appendix A show all of the Persian letters and their various written forms along with their Unicode character codes and examples. Among the letters, there are seven letters which have only two forms: non-joiner and end-joiner. These letters, namely ALEF (‘ا_آ’), DĀL (‘د’), ZĀL (‘ذ’), RE (‘ر’), ZE (‘ز’), ZHE (‘ژ’), and VĀV (‘و’), are the ones that cannot join to the next letter. For instance, the letter DĀL is written as ‘د’ for the non-joiner and

[1]Throughout this dissertation, all of the examples written with the Persian script are read right-to-left.

Table 2.1: List of Persian letters with the same pronunciation

Alphabet	Pronunciation
ط , ت	/t/
ص , س , ث	/s/
ه , ح	/h/
ظ , ض , ز , ذ	/z/
ق , غ	/q/

‘ـد’ for the end joiner. It is necessary to mention that if either of these letters comes in the middle of a word, the next letter will have the same form as it has at the beginning of a word.

Despite Persian uses modifications to the Arabic alphabet, the alphabet is still in a sense better-suited to the Arabic sound system. For instance, the letters ZĀL (‘ذ’), ZE (‘ز’), ZĀD (‘ض’), and ZĀ (‘ظ’) exist in both Persian and Arabic alphabets. Although they still have distinct pronunciations in Arabic, all four letters are pronounced /z/ in Persian; i. e. there are different letters for the same sound in Persian. Table 2.1 shows the letters which have distinct pronunciations in Arabic, but which have acquired identical pronunciations in Persian. Pronouncing these letters identically in Persian increases the number of homophones, such as /savāb/ that can be written either ‘ثواب’ /savāb/ ‘reward’ or ‘صواب’ /savāb/ ‘right action’.

Some letters are also used for representing multiple sounds in Persian, such as VĀV (‘و’) in these examples: /v/ in ‘نانوا’ /nānvā/ ‘baker’, /o/ in ‘دو’ /do/ ‘two’, /ow/ in ‘اوج’ /owǰ/ ‘climax’, and /u/ in ‘روز’ /ruz/ ‘day’. Sometime, this letter is written but not pronounced, such as in ‘خواهر’ /xāhar/ ‘sister’. We can conclude that there is little correspondence between Persian letters and the sounds.

Persian script is written right-to-left, but numbers are written left-to-right, just as in Arabic. In terms of the lexicon, the Persian vocabulary has been greatly influenced by Arabic, and to some extent by French, and a great number of words are borrowed from these languages.

Persian has three short vowels, namely ‘ َ’ /a/, ‘ ُ’ /o/, and ‘ ِ’ /e/, and three long vowels, namely ‘ا ـ آ’ /ā/, ‘و’ /u/, and ‘ی’ /i/. The short vowels are pronounced but usually not written. The Ezāfe morpheme, which is described in more detail in the following section, is always pronounced but rarely written. Table 2.2 shows both short and long vowels including their Unicode encodings and examples.

The Persian script, similar to the Arabic script and contrary to the Latin script, has neither upper nor lower case letters. Therefore, it is not possible to easily identify the beginning of a sentence, and proper or foreign names.

Table 2.2: List of Persian vowels

Vowel	Pronunciation	Unicode	Example	Transliteration	Translation
َ	a	u64E	مَهر	mahr	Mahr
ُ	o	u064F	مُهر	mohr	stamp
ِ	e	u0650	مِهر	mehr	love
آ	ā	u0622	آدم	ādam	man
ا		u0627	مار	mār	snake
و	u	u0648	مور	mur	ant
ی	i	u06CC	میز	miz	table

Moreover, it is not easy to create acronyms.

There are some diacritic characters that are directly borrowed from the Arabic script. Tanvin is one of these characters written as ‘ ً ’ /an/, ‘ ٌ ’ /on/, and ‘ ٍ ’ /en/. Another diacritic character is Tašdid (‘ ّ ’) that causes a consonant to be repeated. The Short Alef ‘'ى’ /ā/ in ‘'حتى’ /hattā/ ‘even’ and ‘ال’ in ‘بالقوه’ /belqovve/ ‘potential’ are two morphological elements borrowed from Arabic. Words are isolated by spaces. Moreover, there exists a so-called ‘pseudo-space’ or ‘zero width non joiner’ as a boundary inside the Persian words. For instance, in the word ‘بین‌المللی’ /beynolmelali/ ‘international’, a pseudo-space is used between the letters ‘ن’ and ‘ا’. If the pseudo-space is not available, the letters will have a joined form, and the word is written as ‘بینالمللی’. If the pseudo-space is replaced by a white-space, this word is considered as two tokens, namely ‘بین’ /beyn/ ‘inter-, between’ and ‘المللی’ /olmelali/ ‘-national’.

Since character codes play the most important roll in typing a text in a computer, standard character codes are used. In 1993, a standard 8 bit code for information exchange was approved for Persian. In 1995, the keyboard standard layout for Persian was approved; and in 1996, the Unicode standard, the 16 bit code, was approved (FahimNiya, 2002).

2.3 Basic Properties of Persian Syntax

In this section, we briefly explain the basic properties of the Persian syntax. Persian has a Subject Object Verb (SOV) constituent order, but the constituent order is relatively free. The language does not make use of gender (Assi, 2004). Furthermore, this language is relatively case-less except for the accusative case in which the post-position (particle) ‘را’ /rā/ is mostly used as a direct object marker or a specificity marker (Karimi, 1999, p. 13), and the genitive case which is expressed by the Ezāfe construction. Persian does not have a definite determiner, but it has a pre-nominal indefinite determiner as ‘یک’ /yek/ ‘a, an’ and a post-nominal indefinite determiner represented as the clitic ‘ی’ /-i/. Definite phrases can freely scramble while scrambling is restricted for indefinite phrases (Karimi, 2003, p. 43).

Ezāfe is an unstressed vowel ‘ ِ ’ /e/ recognized as a morpheme, and it is pronounced but mostly not written. Samvelian (2007) considers Ezāfe as phrasal affix. In addition to its function as a genitive case marker, it causes to link the constituents of a phrase, except the verb phrase. Since Eāfe mostly does not have an overt written form, it is very hard to predict the end of a constituent (Bijankhan et al., 2011). Consequently, it makes the Persian text processing a tough task.
Kahnemuyipour (2000) and Kahnemuyipour (2006) specified that the Ezāfe morpheme can potentially appear in 8 positions. To represent the Persian examples, we follow the Leipzig glossing rules[2] to gloss the Persian examples.

- a noun before another noun:

  (2.1) کیف چرم
  kif-e čarm
  bag-EZ leather
  ‘leather bag’

- a noun before an adjective or adjectival clause:

  (2.2) سگ سیاه
  sag-e siyāh
  dog-EZ black
  ‘black dog’

- a noun before a possessor (noun or pronoun):

  (2.3) کیف مریم
  kif-e maryam
  bag-EZ Maryam
  ‘Maryam’s bag / the bag of Maryam’

- an adjective before another adjective in a noun phrase:

  (2.4) سگ سیاه بزرگ
  sag-e siyāh-e bozorg
  dog-EZ black-EZ big
  ‘big black dog’

- some prepositions before nouns:

  (2.5) زیر کیف
  zir-e kif
  under-EZ bag
  ‘under the bag’

[2]`http://www.eva.mpg.de/lingua/pdf/LGR08.02.05.pdf`

but not:

(2.6) در گنجه *[3]
* dar-e ganǰe
in-EZ closet
'in closet'

- a pronoun before an adjective:

(2.7) من پیر
man-e pir
I-EZ old
'old me'

- first names before last names, but not always:

(2.8) مسعود قیومی
masʔud-e qayyumi
Masood-EZ Ghayoomi
'Masood Ghayoomi'
رضا صادقی
rezā sādeqi
Reza Sadeghi
'Reza Sadeghi'

- a combination of the above:

(2.9) سگ سیاه بزرگ مریم
sag-e siyāh-e bozorg-e maryam
dog-EZ black-EZ big-EZ Maryam
'the big, black dog of Maryam'

It is possible to have compounds with post-nominal adjectives in which Ezāfe is not used (Ghomeshi, 1996), such as 'مادربزرگ' /mādar bozorg/ 'grandmother', 'سیب‌زمینی' /sib zamini/ 'potato', 'آب پرتقال' /āb porteqāl/ 'orange juice', 'آب هویج' /āb haviǰ/ 'carrot juice' (but not 'آب سیب' /ābe sib/ 'apple juice'), and the different words in Persian which all mean 'cousin' in English, such as 'پسرخاله' /persar xāle/. One assumption to remove Ezāfe can be treating the sequences as collocation; i. e. these syntactic constructions are transforming from compound to collocation.

Persian is a subject drop language (Mahootiyan, 1997). It is also possible for an obligatory complement to be dropped in Persian. In other words, the

[3]Based on the convention, the symbol '#' means that the syntactic constructions are syntactically well-formed but semantically improper, and the symbol '*' in the examples means ungrammatical.

subject and the obligatory complement are not found in the surface realization of a sentence. If the subject drops, it can be retrieved from the conjugation of the main verb in the sentence. If a complement drops, the object of the embedded clause that has a surface realization is considered as the object of the matrix clause which does not have a surface realization. In Example 2.10, the subject and a complement in the matrix clause are dropped. The verb 'گفتن' /goftan/ 'say, tell' in the matrix clause requires a subject, an indirect object, and a direct object. The subject is dropped which can be retrieved from the conjugation of the verb. The indirect object is the prepositional phrase 'به من' /be man/ 'to me'. The direct object of the verb in the matrix clause, which is 'حرف' /harf/ 'word' in the embedded clause, is dropped.

(2.10) اگر حرفی داشت باید به من می‌گفت.

agar	harf-i	dāšt	bāyad	be	man	mi-goft
if	word-INDEF	had.3SG	should	to	I	IMPF-said.3SG

'If he/she had a word, he/she should have said [it] to me.'

Topics can also drop in Persian. It is possible that the expletive 'این' /in/ 'this' be dropped, such as Example 2.11a. The consequence of dropping this element is the extrapositon of the other elements in the topic position to the post-verbal position, such as Example 2.11b.

(2.11) a. این که حمید ورشکسته شده‌است دروغ است.

in	ke	hamid	varšekaste	šode-ast	doruq	ast
this	that	Hamid	bankrupt	become-3SG	lie	is

'That Hamid is bankrupted is a lie.'

b. دروغ است که حمید ورشکسته شده‌است.

doruq	ast	ke	hamid	varšekaste	šode-ast
lie	is	that	Hamid	bankrupt	become-3SG

'[It] is a lie that Hamid is bankrupted.'

In Persian, a verb can accept maximum two arguments as objects, but rare cases, mostly as idiomatic expressions, can be found that the verbs' complements are three: one direct object and two indirect objects, such as the verb 'قرار دادن' /qarār dādan/ 'put' in Example 2.12. In this example, the subject is dropped, the particle 'را' /rā/ determines that 'جایزه سیمرغ بلورین' /ǰāyezeye simorqe bolurin/ 'Crystal Phoenix Award' is the direct object, and the two prepositional phrases 'با جایزه خرس طلایی' /bā ǰāyezeye xerse talāyi/ 'with the Golden Bear Award' and 'در یک ردیف' /dar yek radif/ 'in a single rank' are considered as two indirect objects. Elimination of any of these arguments makes the sentence ungrammatical.

(2.12) می‌توان جایزه سیمرغ بلورین را با جایزه خرس طلایی در یک ردیف قرار داد.

mitavān	ǰāyeze-ye	simorq-e	bolurin	rā	bā	ǰāyeze-ye
IMPF.can	award-EZ	phoenix-EZ	crystal	DOM	with	award-EZ

xers-e	talāyi	dar	yek	radif	qarār	dād
bear-EZ	golden	in	one	rank	put	gave.3SG

'One can place the Crystal Phoenix Award with the Golden Bear Award in a single rank.'

Persian has an SOV constituent order, therefore objects should appear before the main verb. Since there are two types of objects (direct and indirect objects) the order of the appearance of the objects is relatively fixed such that the direct object comes before the indirect object. It is possible to freely change the position of the direct and indirect objects and still have a grammatical sentence.

Passive and causative constructions change the number of verbs' arguments. The passive voice in Persian is questionable. Moyne (1974) does not believe that passive constructions exist in Persian, but DabirMoghadam (1986) argues that passive constructions exist in Persian and it is constructed with the light verb 'شدن' /šodan/ 'become'. VahediLangaroodi (1999) considers passivization as the incorporation of a pre-verbal element and the verb 'شدن' /šodan/ 'become' to create a compound verb.
In causative constructions, the suffixes 'ـانیدن' /-ānidan/ and 'ـاندن' /-āndan/ are added to the present stem of verbs to create the infinitive forms of causative verbs for past and present tenses, respectively. As an example, the past and present causative verbs 'خوابانیدن' /xābānidan/ 'have made sleep' and 'خواباندن' /xābāndan/ 'make sleep' are constructed of adding the causative affixes to 'خواب' /xāb/ 'sleep', the present stem of the verb 'خوابیدن' /xābidan/ 'sleep'.

There are simple and compound verbs in Persian. The compound verbs are constructed of a pre-verbal element, such as a prepositional phrase, a noun, or an adjective, and a light verb (DabirMoghaddam, 1997).
KarimiDoostan (1997) divided the light verbs to three classes: (a) *stative*, such as 'داشتن' /dāštan/ 'have'; (b) *transition*, such as 'شدن' /šodan/ 'become; (c) *initiatory*, such as 'کردن' /kardan/ 'do'. The author further added that there are three classes of pre-verbal nouns: (a) the *non-predicative* nouns which are concrete nouns, such as 'گوش' /guš/ 'ear' in 'گوش کردن' /guš kardan/ 'listen'; (b) the *process* nouns, such as 'راهنمایی' /rāhnemāyi/ 'advice' in 'راهنمایی کردن' /rāhnemāyi kardan/ 'guide'. The process nouns can co-occur with the post nominal indefinite determiner 'ی' /-i/ and the direct object marker 'را' /rā/; (c) the *verbal nouns*, such as 'انجام' /anǰām/ 'perform' in 'انجام دادن' /anǰām dādan/ 'perform'. The verbal nouns can co-occur with adjectives.
It should be added that the pre-verbal elements are separable from the verbal elements such that the pre-verbal elements might contain other elements

as their argument structure[4] and these pre-verbal elements dominate these elements. Furthermore, it is possible for the argument structure of a compound verb to be different from the argument structure of the light verb used in the compound form. The simple and compound word forms in the lexicon are not limited to the verbs only, but cases can also be found for adjectives, adverbs, conjunctions, WH-question words (SharifiAtashgah and Bijankhan, 2009), and prepositions (AbolhassaniChime, 2006; AbolhasaniChime and Ghayoomi, 2006).

To construct polar questions, the interrogative word 'آیا' /āyā/ 'whether' is added to the front of a declarative sentence, and the placement of all elements remains unchanged. In interrogative sentences with WH-question words, they do not enforce any obligatory extraposition to the front of a sentence and the elements of the sentence can remain in-situ.

The past, present, and future tenses exist in Persian. The verbs have the present and past stems. The past stem of a verb is achieved simply by removing the infinitive marker 'ـن' /-an/ from the end of a verb, while the present stem is irregular. Verbs have four aspects: (a) *present*, (b) *present perfect*, (c) *progressive* (imperfect) such that the prefix 'می‌ـ' /mi-/ is followed by a verb either in the past or present tense, and (d) *past perfect* such that the past participle form of the verb is followed by the auxiliary verb 'بودن' /budan/ 'be'. To construct the future tense, the auxiliary verb 'خواستن' /xāstan/ 'will, would' comes before either the past or present stem of the main verb. Using either of the stems after the auxiliary 'خواستن' /xāstan/ 'will, would' depends on the aspect of the auxiliary. If the auxiliary appears in present aspect, the past stem of the verb is used. Furthermore, if the auxiliary appears in imperfect aspect, the present stem of the verb is used. In terms of mood, *subjunctive* and *imperative* forms exist in Persian, and the subjunctive mood is frequently used.

Persian has three types of modal verbs: (a) it conjugates and there is an agreement between the subject and the verb in terms of the person and number, such as 'توانستن' /tavānestan/ 'can, could' in Example 2.13, (b) it only conjugates for the second and third persons singular, such as 'بایست' /bāyest/, 'بایستی' /bāyesti/, 'می‌بایست' /mibāyest/, 'می‌بایستی' /mibāyesti/ which all mean 'have to, has to, had to' in Example 2.14; (c) it is impersonal and it does not conjugate, such as 'باید' /bāyad/ 'must, should', and 'شاید' /šāyad/ 'might', 'می‌شود' /mišavad/ 'may', 'می‌شد' /mišod/ 'might', and 'بشود' /bešavad/ 'might' in Examples 2.15 and 2.16 (MeshkatoDini, 2001, pp. 74–75); (Tabibzadeh, 2012, pp. 77–78).

[4]The argument structure refers to the syntactic and semantic configuration of a lexical item.

(2.13) من توانستم به تهران بروم.
man tavānest-am be tehrān be-rav-am
I could-1SG to Tehran SUBJ-go-1SG
'I could go to Tehran.'

(2.14) من بایست به تهران بروم.
man bāyest be tehrān be-rav-am
I have_to.3SG to Tehran SUBJ-go-1SG
'I have to go to Tehran.'

(2.15) من باید به تهران بروم.
man bāyad be tehrān be-rav-am
I must to Tehran SUBJ-go-1SG
'I must go to Tehran.'

(2.16) من شاید به تهران بروم.
man šāyad be tehrān be-rav-am
I might to Tehran SUBJ-go-1SG
'I might go to Tehran.'

Gholamalizadeh (1999) recognized five main lexical categories as constituents' heads in Persian (noun, preposition, adjective, adverb, and verb) which construct five phrasal categories: noun phrases which contain pre-nominal and post nominal elements, prepositional phrases, adjectival phrases, adverbial phrases, and verb phrases. The Persian pre-nominals are a superlative adjective, a demonstrative, an indefinite, interrogative, or exclamatory word, a number (ordinal number (type I)[5], or cardinal number), classifier, title (type I)[6], and the indefinite determiner 'یک' /yek/ 'a, one'. The post-nominal elements are a phrase (nominal , adjectival, or prepositional), a title (type II), a relative clause, the indefinite determiner 'ی' /-i/ as a clitic, an adverb (place, time, or quantity), and an ordinal number (type II).

2.4 Language Resources for Persian

There are a number of corpora for Persian that are developed for special purposes and they vary in size and domain. Most of the developed corpora for Persian are texts, but speech corpora exist for Persian as well. In this section, we describe the existing text corpora for Persian along with their applications.

The Farsi Linguistic Data Base (FLDB) is a linguistic corpus which contains 3 million words in the ASCII format released by Assi (1997) at the Institute for

[5]There are two types of ordinal numbers in Persian. Type (I) appears before a noun, such as 'اولین' /avvalin/ 'the first', but type (II) appears after a noun, such as 'اول' /avval/ 'first'.

[6]There are two types of titles in Persian. Type (I) appears before a noun, such as 'آقای' /āqāye/ 'Mr.', but type (II) appears after a noun, such as 'خان' /xān/ 'khan'.

Humanities and Cultural Studies, Tehran, Iran. The most recent version of the database in 1256 character encoding is named the Persian Linguistic Data Base (PLDB), and it includes more than 56 million words (Assi, 2005). This database is comprised of contemporary literary books, articles, magazines, newspapers, parliament laws and regulations, transcriptions of news, reports, and telephone speeches and it is employed for the lexicography purpose (Assi, 2006). In this database four labels, such as phonetic tag, POS tag, semantic role label, and lemma, are assigned to each word. The POS tag set is a set of 44 tags (Assi and HajiAbdolhosseini, 2000). Phonetic tags are assigned semi-automatically, and semantic role labeling and lemmatization are done manually. A small portion of this corpus is fully annotated with these four types of labels.

The online archive of the Hamshahri newspaper has been used for creating several Persian corpora. Ghayoomi (2004) developed a corpus from 6 months of the Hamshahri newspaper with more than 6.5 million words used for language modeling. Darrudi et al. (2004) developed another corpus from 4 years of the Hamshahri newspaper which contains more than 37 million words. AleAhmad et al. (2009) standardized the Hamshahri corpus according to TREC[7] specifications (Craswell and Hawking, 2004). This data set, which was developed in the DataBase Research Group Lab at the University of Tehran, Tehran, Iran, contains 65 standard query topics along with 6,500 relevance judgments. The data set is mainly employed for text retrieval. Oroumchian et al. (2004) comprised a 2.5 million words text collection that contains laws and regulations passed by the Iranian Parliament. The Shiraz Corpus is a bilingual parallel corpus that consists of 3,000 Persian sentences with the corresponding English sentences. The corpus is collected from the Hamshahri newspaper. All sentences are manually translated at the Computing Research Lab at New Mexico State University (Amtrup et al., 2000).

Peykareh is a balanced corpus for contemporary Persian. This corpus is developed at the Research Center for Intelligent Signal Processing, Tehran, Iran, and it contains approximately 100 million words (Bijankhan et al., 2004; Bijankhan, 2007; Bijankhan et al., 2011). This corpus consists of newspapers, books, magazines, articles, technical reports, as well as transcription of dialogs, monologues, and telephone speech employed for the language modeling purpose. Almost 10 million words of this corpus are POS tagged semi-automatically based on the EAGLES guidelines (Leech and Wilson, 1999) such that a small fraction of data is tagged manually to train a statistical POS tagger with 85% accuracy, and the rest is tagged automatically and corrected manually. Based on the EAGLES guidelines, there is a hierarchy on the assigned POS tags. As a result, the first tag which expresses the main syntactic category of a word is followed by

[7]`http://trec.nist.gov/`

morpho-syntactic and semantic features. The POS tag set used for annotating this fraction of data is represented in Tables B.1 and B.2 in Appendix B. A small fraction of this annotated data set is called the Bijankhan Corpus[8] (Bijankhan, 2004), which is freely available online. The Bijankhan Corpus contains around 2.5 million word tokens. For our study, we select 1,000 sentences from this corpus. This small fraction of data is used for starting the first phase of treebanking that is described in Chapter 5.

2.5 Challenges in Persian Text Processing

When processing a Persian text corpus, there are problems that one might encounter because of the specific properties of the Persian script. These problems can be solved automatically or manually. In this section, the most frequent problems are listed and explained.

2.5.1 Encoding Issues

The control characters[9] for both Persian and Arabic are very similar to each other, but in terms of other characters, there are differences between the two. One difference is having four more letters in the Persian script that do not exist in the Arabic script. The other one is that the written texts sometime employ Arabic characters in addition to the range of the Unicode characters designed for Persian. Hence, the letters KĀF ('ک') and YE ('ی') can be expressed by either the Persian Unicode encoding (u06a9 for 'ک' and u06CC for 'ی') or by the Arabic Unicode encoding (u0643 for 'ك' and u064A for 'ي') (Megerdoomian, 2004). Most of the operating systems have Arabic codes as their default character encoding and only the codes for PE ('پ'), CHE ('چ'), ZHE ('ژ'), KĀF ('ک'), GĀF ('ک'), and YE ('ی') are added to be compatible with Persian. This results in mixing the character codes of the two languages. Since character codes play the roll for sorting and running a query, the mixture of Arabic and Persian character codes makes processing of a Persian text difficult.

Since these characters are limited, the problem of mixture of character encodings can be solved by converting these letters to their normalized forms. It should be added that there is a corresponding mapping of the Arabic 1256 encoding to a particular Unicode encoding based on the ISIRI 6219 standard for Persian. This solves the problem of character encoding, but not sorting. The sorting problem still exists in the operating systems, and the problem should be resolved in individual applications. The normal sorting of Persian letters is as

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

[9]This set of characters does not print, and it does not have a written representation.

follows:
الف، ب، پ، ت، ث، ج، چ، ح، خ، د، ذ، ر، ز، ژ، س، ش، ص، ض، ط، ظ، ع، غ، ف، ق، ک، گ، ل، م، ن، و، هـ، ی.[10]
Applying the sorting of Arabic letters on Persian will make two changes on the order of Persian letters: (a) the positions of the letters VĀV ('و') and HE 'هـ' are swapped as is in the Arabic alphabet, (b) the six letters, namely PE ('پ'), CHE ('چ'), ZHE ('ژ'), KĀF ('ک'), GĀF ('گ') and YE ('ی'), which do not exist in the Arabic script appear after the Arabic alphabet. So that, the following incorrect order is what results:
الف، ب، ت، ث، ج، ح، خ، د، ذ، ر، ز، س، ش، ص، ض، ط، ظ، ع، غ، ف، ق، ل، م، ن، هـ، و، پ، چ، ژ، ک، گ، ی.[11]

2.5.2 Internal Word Boundaries

One of the biggest issues in processing Persian texts is the internal word boundary that should ideally be represented with the so-called 'pseudo-space'. In many texts, the correct usage of pseudo-space is not given much importance, and often either a white-space is inserted, or the internal word boundary is completely ignored. Using a white-space instead of the pseudo-space causes a problem in the tokenization, and it raises the 'multi unit token' problem. Furthermore, this problem has a negative impact on the frequency distribution of words in text processing. For instance, using a white-space in typing the word 'بین‌المللی' /beynolmelali/ 'international' as 'بین المللی' splits this word to two separate tokens, namely 'بین' and 'المللی'. Nevertheless, ambiguity in the tokenization makes this problem more severe since it has a negative effect on POS tagging for assigning the word a proper POS tag. The tokenization of the strings 'وبا' and 'مادر' can be either as one token to have 'وبا' /vabā/ 'cholera' and 'مادر' /mādar/ 'mother', or two tokens to have 'و با' /va bā/ 'and with' and 'ما در' /mā dar/ 'we in'.

Moreover, optionality of the internal word boundary raises problems in the analysis of detached affixes and morphemes, such as 'می‌ـ' /mi-/ (present and past imperfective morpheme), 'هم‌ـ' /ham-/ (prefix '-mate'), 'بی‌ـ' /bi-/ (prefix '-less'), 'ـها' /-hā/ (plural morpheme), 'ـای' /-i/ (post-nominal determiner morpheme), 'ـتر' /-tar/ (comparative suffix), or 'ـترین' /-tarin/ (superlative suffix). The inflectional morphemes can appear either bound or separated as represented in Table 2.3. Bound morphemes join the host with a pseudo-space. Separated morphemes contain an intervening white-space. Although usage of

[10]ALEF, BE, PE, TE, SE, JIM, CHE, HE, KHE, DĀL, ZĀL, RE, ZE, ZHE, SIN, SHIN, SĀD, ZĀD, TĀ, ZĀ, EYN, GHEYN, FE, GHĀF, KĀF, GĀF, LĀM, MIM, NUN, VĀV, HE, YE

[11]ALEF, BE, TE, SE, JIM, HE, KHE, DĀL, ZĀL, RE, ZE, SIN, SHIN, SĀD, ZĀD, TĀ, ZĀ, EYN, GHEYN, FE, GHĀF, LĀM, MIM, NUN, HE, VĀV, PE, CHE, KĀ, GĀF, YE

Table 2.3: Different types of word boundaries for affixes

Affix	White-space	Pseudo-space	Attached	Transliteration	Translation
می–	می گوید	می‌گوید	میگوید	miguyad	says
هم–	هم کلاسی	هم‌کلاسی	همکلاسی	hamkelāsi	classmate
بی–	بی نیاز	بی‌نیاز	–	biniyāz	needless
–ها	پول ها	پول‌ها	پولها	pulhā	monies
–ای	خانه ای	خانه‌ای	–	xāneʔi	a house
–تر	بزرگ تر	بزرگ‌تر	بزرگتر	bozorgtar	bigger
–ترین	بزرگ ترین	بزرگ‌ترین	بزرگترین	bozorgtarin	biggest

Table 2.4: Different word boundaries for derived and inflected words

White-space	Pseudo-space	Attached	Transliteration	Translation
بین المللی	بین‌المللی	–	beynolmelali	international
زبان شناسی	زبان‌شناسی	زبانشناسی	zabānšenāsi	linguistics
کتاب سرا	کتاب‌سرا	کتابسرا	ketābsarā	book-house
دانش آموز	دانش‌آموز	–	dānešāmuz	student
علاقه مند	علاقه‌مند	علاقمند	alāqemand	interested

pseudo-space is the standard form for such cases, the other forms are frequently used in varying combinations depending on the style of the person who compiles the text.

Apart from affixes, there are other categories that have this problem, especially in derived and inflected words. Table 2.4 shows a number of samples that are usually written with or without white-space though they should be written with pseudo-space to be considered as a single token.

Orthographically, clitics are always attached to their hosts. Samvelian (2007) recognized clitics as phrasal affixes. The required changes on clitics are discussed in more detail in Section 5.4.1.

The white-space problem and the tokenization problem make the processing of complex words consisting of separate lexical items a tough task. This problem is also called the ‘multi token’ problem, and it exists at various levels (Megerdoomian, 2004; SharifiAtashgah and Bijankhan, 2009). If the ‘multi token’ presents a concept, it is called ‘multi unit token’, such as compound nouns that are mostly linked together with the Ezāfe morpheme, compound prepositions which are constructed based on the incorporation of two simple prepositions or a simple preposition and a noun (AbolhassaniChime, 2006; AbolhasaniChime and Ghayoomi, 2006), compound adjectives, compound adverbs, compound conjunctions, or numerals in the alphabetical form. If the ‘multi token’ does not present a concept and the containing elements are morphologically independent, then it is called ‘multi token unit’. One reason that

Table 2.5: Different kinds of word boundaries for multi token words

Type	White-space	Pseudo-space	Attached	Transliteration	Translation
Noun	ماشین لباسشویی تخم مرغ	ماشین‌لباسشویی تخم‌مرغ	– –	māšinelebāsšuyi toxmemorq	washing machine egg
Preposition	به شیوه بعد از	به‌شیوه –	بشیوه بعداز	bešiveye baʔdaz	like after
Adjective	سنگین وزن آبی رنگ	سنگین‌وزن آبی‌رنگ	– –	sanginvazn ābirang	heavy blue
Adverb	هر از چند گاه به طور دقیق در غیر این صورت	– به‌طوردقیق درغیراین‌صورت	هرازچندگاه بطوردقیق درغیراینصورت	harazčandgāh betoredaqiq darqeyreʔinsurat	rarely exactly otherwise
Conjunction	به محض این که به این شکل که	به‌محض‌این‌که به‌این‌شکل‌که	– –	bemahzeʔinke beʔinšeklke	as soon as such taht
Numeral	صد و بیست و سه هزار	صدوبیست‌وسه‌هزار	–	sadobistosehezār	123000
Determiner-Noun	این کار	این‌کار	اینکار	in kār	this work
Pronoun-Post-position	او را	–	اورا	u rā	him/her
Conjunction-Preposition	و با	–	وبا	va bā	and with
Pronoun-Preposition	ما در	–	مادر	va bā	we in
Numeral-Noun	۱۲۳ بشکه	–	۱۲۳بشکه	sadobistose boške	123 barrels

this problem occurs is that certain words end with the non-joiner letters (‘آ ـ ا’, ‘د’, ‘ذ’, ‘ر’, ‘ز’, ‘ژ’, and ‘و’) that do not attach to the next word. Another source of the problem is numerals written in digit format and without inserting a white space the numerals are attached to the next word. The first six rows of Table 2.5 represents samples for multi unit tokens; and the last five rows are samples for multi token units.

2.5.3 Writing Styles

There exist three kinds of language varieties: standard, super-standard, and sub-standard. The standard language both for oral and written is the one used as the official language in press, mass media, formal communications and correspondences, etc. The super-standard language which has more cultural and artistic values is used in literary texts and scientific text books. The sub-standard language is used in Short Message Services (SMS) and blogs. Slangs are sub-branches of this variety. We cannot always have a rigid and clean border between these varieties as there is a possibility to have an amalgamation of them within a text. The reason is that the writer might shift from one style to another. This language variability adds complexity to text processing. For instance, the word ‘if’ can be written as ‘اگر’ /agar/ in standard and super-standard texts, ‘گر’ /gar/ in super-standard texts, and ‘اگه’ /age/ in sub-standard texts.

2.5.4 Linguistic Creativity

Aside from the normal language changes, recent communication technologies like SMS have added creativity to the written form of the words in Persian. Typing complete words on a mobile phone, especially compound words that contain several characters, is very time-consuming for the users, and pressing less keys is more convenient. To speed up messaging, users might create a word form from the original one which has less number of characters. For example, the verb 'زنگیدن' /zangidan/ is used instead of the compound verb 'زنگ زدن' /zang zadan/ 'call' in SMS texts. It should be added that this phenomenon is not limited to SMS texts as it is entered to the chat-rooms and blogs as well.

Additionally, some writers of literary and novel books that belong to the super-language class use their own writing style, and they make minor changes to the words borrowed from Arabic without any changes to the pronunciation, meaning, or syntactic function. For example the words 'حتی' or 'حتیٰ' /hattā/, and 'حتماً' /hatman/ 'certainly', which are the standard orthographical forms, are written as 'حتا' and 'حتمن'.

2.5.5 Homographs and Homonyms

Like other languages, Persian also has ambiguities in its lexicon. However, because of two important properties of the Persian script, the number of homographs and homonyms is surprisingly high. One property is that usually short vowels are not written in Persian. A Persian morphological analyzers should deal with this problem to analyze and disambiguate them. As an example, the homograph 'کند' can be pronounced any of these along with their POS tags: /kand/ 'picked' (Verb, Past Tense), /kanad/ 'picking up' (Verb, Present Continuous Tense), /konad/ 'doing' (Verb, Present Tense), /kond/ 'slow' (Adverb), and /kond/ 'blunt' (Adjective). This property has a negative impact on the tokenization to extract the word frequency, because these five words will be treated as one. Bijankhan et al. (2011) defined syntactic patterns to classify the homographs.

The other property is that capitalization does not exist in Persian, therefore proper nouns cannot be simply distinguished from common nouns, unlike English. This makes the automatic recognition of the named entities a tough task. For example, the word 'آذر' /āzar/ is either the name of the 9th month in the Persian calendar, a woman's name, or a common noun that means 'fire'.

2.5.6 Borrowed Diacritic Characters from Arabic

There are four diacritic Arabic characters that add problems to Persian text processing, namely Tanvin, Tašdid, Hamze, and Short Alef. Some typists use Tanvin and some do not. This variability results in two entries in the lexicon for a single token, such as 'فعلاً' /feʔlan/ 'yet' and 'فعلا'. This problem can be a source of ambiguity for cases like 'جدا' that can be either 'جدا' /ǰodā/ 'separate' or 'جداً' /ǰeddan/ 'really'.

Tašdid ' ّ ' can be written or ignored, such as 'معلّم' and 'معلم' /moʔallem/ 'teacher'. Its missing can be another source of ambiguity. For instance, the word 'بنا' could be either 'بنا' /banā/ 'building, base' or 'بنّا' /bannā/ 'bricklayer'.

Hamze is an Arabic character which optionally appears in Persian words. This character can be written as 'ء' at the end of a word, such as 'املاء' /emlā/ 'dictation', or it can be ignored, such as 'املا'. Hamze is written as 'أ', 'ئ', or 'ؤ' in the middle or end of words. There is a debate among Iranian linguists on the subject of Hamze. The first group do not believe in the existence of Hamze in Persian and they use the corresponding long vowel ('ا', 'ی', and 'و') instead of Hamze, such as 'مساله' /masale/ 'problem', 'رییس' /reyis/ 'boss', and 'مومن' /momen/ 'believer'. Alternatively, these words can be written as 'مسئله' or 'مسأله' /masʔale/, 'رئیس' /reʔis/, and 'مؤمن' /moʔmen/ based on the assumption of the second group who believe in the existence of Hamze in Persian. Having Hamze, removing it, or replacing it with its associated long vowel is ambiguous and undecided. For example, the word 'mirror' has three different written forms: with Hamze as 'آئینه' /āʔine/, without Hamze as 'آینه' /āyene/, and with 'ی' replacement as 'آیینه' /āyine/.

Similar to other borrowed characters, some typists use the Short Alef 'ٰی' and some do not, such as 'even' that can be written as either 'حتٰی' or 'حتی'.

2.5.7 Various Orthographical Forms for Words

Persian orthography is not completely rule-based and standardized. Even though the Persian Academy of Language and Literature (2005) has published a book for this purpose, a list of exceptions is presented after the rules. Shamsfard et al. (2010) designed a software to normalize a Persian text according to the Persian Academy of Language and Literature (2005). Seraji et al. (2012b) and Sarabi et al. (2013) also tried to normalize a Persian text automatically.

Besides the multi token problems, different typing styles also add further problems. One of these problems is caused by the nature of 'ا – آ'. As presented in Tables A.1 and A.2 in Appendix A, the letter 'آ' appears quite often at the very beginning of the words, very rarely in the middle of the words; but the letter 'ا' pronounced /ā/ appears only in the middle or at the final positions.

However, since these two letters are pronounced the same as /ā/, some typists use 'ا' instead of 'آ'. For instance, the word 'آرام' /ārām/ 'calm' is typed as 'ارام'. Mixing this problem with the variability of using Hamze causes various orthographical forms of words. For instance, the word 'American' is spelled as 'امریکائی' /emrikāʔi/, 'امریکایی' /emrikāyi/, 'آمریکائی' /āmrikāʔi/, and 'آمریکایی' /āmrikāyi/. To overcome such inconsistencies, Ghayoomi (2004) used the highest frequency of the orthographical forms as a guide to decide on the default spelling and replaced the various spellings with the selected candidate spelling.

In Persian, Ezāfe is represented in two forms based on the syllabic restrictions in Persian. In one form, it appears as ' ِ ' /e/ after a consonant, though it is usually not written, such as in 'کیفِ مریم' /kife maryam/ 'Maryam's book'. The other form is when Ezāfe appears after a vowel in which the intermediary morpheme 'ی' /y/, a consonant, is inserted to follow the syllabic restrictions in Persian. Depending on whether the previous letter is a consonant or a vowel and whether white-space or pseudo-space is used result in four types of writing formats for Ezāfe. In two of the formats, Ezāfe is written with the intermediary morpheme 'ی' /y/ along with a white-space or pseudo-space at the end of the word, such as 'خانه‌ی' or 'خانه ی' /xāneye/ 'the house of'. Another written format is writing Ezāfe in the form of 'ـۀ' at the end of a word like 'خانۀ' /xāneye/. The last form is writing no Ezāfe in either of the above formats, such as 'خانه'.

2.5.8 Foreign Words

Since the Persian alphabet differs considerably from the Latin script, writing foreign words in Persian is not straightforward. There is no standard method for writing these words, and they may be written in different forms depending on how individual writers choose to adapt them to Persian phonology, such as the word 'intermediate' which can be written as 'اینترمدییت' /intermediyet/, 'اینترمدیت' /intermediyet/, 'اینترمیدیت' /intermidiyet/, and 'اینترمیدییت' /intermidiyet/.

2.6 Summary

In this chapter, the basic properties of Persian, its syntactic properties, and the available language resources for this language were described. Additionally, most of the common problems one might face in processing a Persian text corpus were mentioned. The source of the problems could be the Persian script mixed with the Arabic script, the Persian orthography, the typing style used in the compilation of a text, the characters' encoding in the operating systems, linguistic styles, and creativity in the written form of the words.

Chapter 3

Grammar Formalisms, Treebanking, and Parsing

3.1 Introduction

In this chapter, we provide background and the related studies in the literature. This chapter contains five sections. In Section 3.2, some of the well-known grammar formalisms that have applications in computational linguistics are briefly described, and their basic concepts are introduced. The HPSG formalism and the previous studies on the Persian HPSG are also discussed in this section. In Section 3.3, the notion of treebanking is defined, and the previous studies on major treebanks in general and the Persian treebanks in particular are described. Section 3.4 has a focus on parsing, and it deals with parsing methods, evaluation metrics, and the previous studies on parsing Persian. Section 3.5 summarizes this chapter.

3.2 Grammar Formalisms

A formal language is defined as "a set of strings [such that] each string [is] composed of symbols from a finite symbol-set called alphabet" (Jurafsky and Martin, 2000, p. 39). This formal language is used for modeling a natural language. In the 1950s, Chomsky (1957) proposed a formal language to describe the natural language grammar formally. In Chomsky's view, a language is constructed of a finite set of grammar rules that are used recursively to create sentences. These grammar rules describe a chain of words such that the grammar rules are organized hierarchically to provide the syntactic analysis of a sentence. Constituents are the smallest syntactic units that are constructed based on the grammar rules.

A set of labels are used for labeling the constituents. Since one word, called the 'head', plays the most important role to construct a constituent, the syntactic category of the head word is projected to its constituent label.

This way of analyzing a natural language caught the attention of researchers in other areas, and they proposed more advanced formalisms to describe the natural language more precisely. The consequence of their studies is proposing various grammar formalisms with their own properties. There are two major ways to provide the syntactic analysis of a sentence: (a) based on Chomsky's generative grammar which is topological (constituency-based), and (b) tecto-grammatical (dependency-based) in which only the syntactic relations are considered.

A grammar formalism can serve as the backbone when building a treebank. One advantage of such an approach, as will be described in Section 3.3, is that the development of a new treebank based on another grammar formalism can be done through a conversion strategy, which is faster than its development from scratch. As a result, we briefly introduce in this section the properties of well-known grammar formalisms that have applications in computational linguistics.

3.2.1 Constituent-based Analysis

The major grammar formalisms of the constituent-based analysis that have relevance to computational linguistics are Phrase Structure Grammar, Categorial Grammar, Tree Adjoining Grammar, and Lexical Functional Grammar, which are all briefly described in this section.

Phrase Structure Grammar

Phrase Structure Grammar (PSG) is the primary grammar formalism proposed by Chomsky (1957). This grammar is a generative device that is constructed of a finite set of rewriting grammar rules containing terminals (words) and non-terminals (constituents). The interaction between the grammar rules creates an infinite number of sentences. Grammar Rules 3.1 generate Examples 3.2a-f but not Examples 3.2g and 3.2h which violate the given grammar rules. Example 3.3 provides the linear analysis of the syntactic structure of Example 3.2a in which the sentence is bracketed and the constituent labels are assigned. Figure 3.1 represents this analysis graphically as a tree diagram. Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), and HPSG (Pollard and Sag, 1994) are the two advanced versions of the original PSG. HPSG is described in more detail in Section 3.2.3.

(3.1)

$$
\begin{array}{lll}
\text{S} \rightarrow \text{NP VP} & \qquad & \text{Det} \rightarrow \text{the} \\
\text{NP} \rightarrow \text{Det N} & & \text{N} \rightarrow \text{boy} \mid \text{book} \mid \text{apple} \\
\text{VP} \rightarrow \text{V NP} & & \text{V} \rightarrow \text{studies} \mid \text{eats}
\end{array}
$$

(3.2) a. The boy eats the apple.
b. The boy studies the book.
c. # The book eats the boy.
d. # The book studies the apple.
e. # The apple eats the boy.
f. # The apple studies the book.
g. * The eats boy the apple.
h. * Eats the boy the apple.

(3.3) [[The$_{Det}$ boy$_{N}$]$_{NP}$ [eats$_{V}$ [the$_{Det}$ apple$_{N}$]$_{NP}$]$_{VP}$]$_{S}$

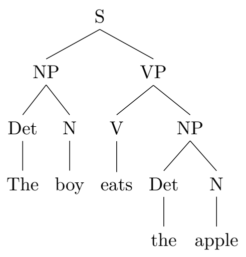


Figure 3.1: Tree diagram of Example 3.2a

Categorial Grammar

Categorial Grammar (CG) was first proposed by Adjukiewicz (1935) and it was modified later by Lambek (1958) and others. A categorial grammar is constructed based on two components: (a) a categorial lexicon in which words are associated with their syntactic and semantic categories where the words' combinatorial potentials are defined, and (b) the inference rules which are the functions to determine how and in which direction the arguments should follow based on the constituents' symbols (Steedman, 1993). This formalism, which is mostly semantically oriented for syntactic combination, is motivated by Montague's principle of compositionality (Montague, 1974). A category can be a basic category, such as N, or a functor category, such as a transitive verb that requires a function to map its required arguments. A functor category can be defined by the left-right operators (X/Y and X\Y). X/Y means that if an element combines with the element Y on its right, it will produce X; and X\Y means that the existence of an element on the left of Y will produce X. Figure 3.2 illustrates the derivation of Example 3.2a based on CG.

Combinatory Categorial Grammar (CCG) is a specific version of CG, and it differs slightly from CG in the way directionality is denoted (Steedman, 1993). It needs to be added that it is possible to enrich the functions with lambda expressions for composing the elements.

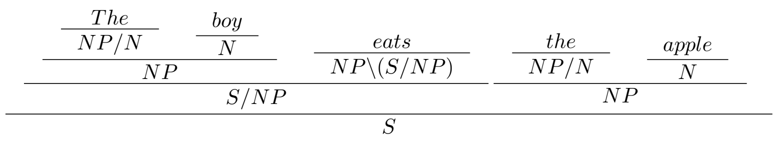


Figure 3.2: CG inference of Example 3.2a

Tree Adjoining Grammar

Tree Adjoining Grammar was proposed by Joshi (1985). The main property of this formalism is that trees, rather than grammar rules, play the most important role. There are two types of trees in this formalism: (a) the initial tree which represents the lexical knowledge as a simple tree structure, and (b) the auxiliary tree which allows for recursion. Two operations are used for combining the two types of trees, namely substitution, and adjunction. The nodes of the trees that allow substitution are represented with the arrow-down symbol (↓), and the nodes that allow adjunction are marked with a star (*). In the extended version of this formalism, the trees are lexicalized such that at least one node of a tree is anchored to a lexical item; as a result it is called Lexicalized Tree Adjoining Grammar (LTAG) (Schabes, 1990).
The initial trees which allow substitution in Example 3.2a are represented in Figure 3.3. The initial tree in Figure 3.4 is used for adjunction in a sentence, such as ‘The boy eats the apple greedily’.

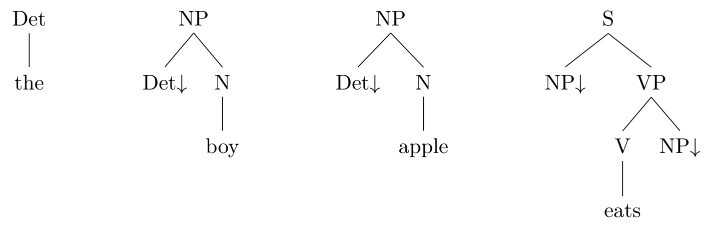


Figure 3.3: LTAG initial trees used for the substitution operation

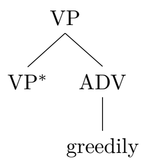


Figure 3.4: LTAG initial tree used for the adjunction operation

Lexical Functional Grammar

Lexical Functional Grammar (LFG) was introduced by Bresnan and Kaplan (1982). This formalism is a direct map between semantic argument structures

and surface grammatical relations (Kaplan and Bresnan, 1995). This formalism uses a set of syntactic rules and a set of lexical entries to provide a mapping between semantic functions and the surface grammatical functions. LFG consists of several levels for describing the syntactic relations. Two of the most important levels are: (a) the constituent structure (*c*-structure) represented by trees, and (b) the functional structure (*f*-structure) represented by attribute-value matrices (AVM). The LFG-style Grammar Rules 3.4 enriched with functional annotations represent the *c*-structure of Example 3.2a.
The functional annotations are represented with the up and down arrows to symbolize the meaning graphically. The up arrow points to the mother node that is matched to a node in the *f*-structure with the defined function. The up arrow in (↑SUBJ)=↓ represented in Grammar Rules 3.4 means that the mother node NP attaches to the SUBJ node in the *f*-structure. The down arrow points to the current node, and it corresponds the matching node in the *f*-structure to this node. The syntactic and semantic properties of the lexical items in Example 3.2a are defined in Example 3.5. As stated above, the provided information in the *c*-structure should map to the semantic functions in the *f*-structure. Figure 3.5 represents the relevant *f*-structure of Example 3.2a.

(3.4)

S →	NP	VP
	(↑ SUBJ)=↓	↑ = ↓
VP →	V	NP
	↑ = ↓	(↑ OBJ)=↓
NP →	Det	N
	↑ = ↓	↑ = ↓

(3.5)

Det : the
(↑SPEC)=‘the’

N : boy | apple
(↑NUM)=SG | (↑NUM)=SG
(↑PRED)=‘boy’ | (↑PRED)=‘apple’

V : eats
(↑PRED)=‘eat < SUBJ , OBJ >’
(↑NUM)=SG
(↑PERS)=3
(↑TENSE)=PRESENT

3.2.2 Dependency-based Analysis

Besides the constituent-based generative grammars, dependency-based representations (Mel’čuk, 1988) have also been proposed. To display the data with

$$\begin{bmatrix} \textsc{pred} & \textit{`eat}\left\langle(\uparrow \textsc{subj}), (\uparrow \textsc{obj})\right\rangle\textit{'} \\ \textsc{num} & \textit{SG} \\ \textsc{pers} & \textit{3} \\ \textsc{tense} & \textit{PRESENT} \\ \textsc{subj} & \begin{bmatrix} \textsc{spec} & \textit{the} \\ \textsc{num} & \textit{SG} \\ \textsc{pred} & \textit{`boy'} \end{bmatrix} \\ \textsc{obj} & \begin{bmatrix} \textsc{spec} & \textit{the} \\ \textsc{num} & \textit{SG} \\ \textsc{pred} & \textit{`apple'} \end{bmatrix} \end{bmatrix}$$

Figure 3.5: *f*-structure of Example 3.2a

dependencies, the data is transformed to Directed Acyclic Graphs (DAG) with labeled edges that determine the type of the dependency relations. Dependency Grammar (DepG) has various approaches to define the relations, such as having crossed edges (non-projectiveness) or having more than one incoming edge on a node for structure sharing (re-entrancy) (Nivre, 2005a). Moreover, this sort of analysis is a soft representation of the linguistic analysis, since in the analysis there is no hierarchical constituent structures and the number of edges in the graphs is equal to the number of tokens in the sentence (Hajič, 1998). Empty categories like traces or multi-word expressions (either idiomatic expressions or multi tokens, such as 'to walk' or 'in addition to') are not dealt with in this sort of analysis. As a result, these properties might be the reason why dependency representation is used for languages with less configurational word order than the languages with more restricted constituent order like English. Figure 3.6 represents the dependency analysis of Example 3.2a.

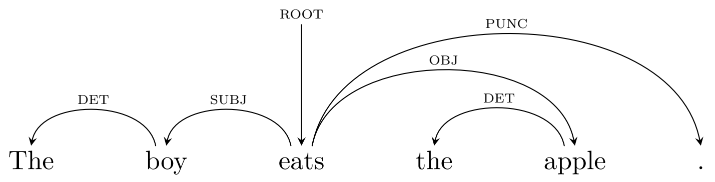


Figure 3.6: Dependency representation of Example 3.2a

Link Grammar (LinkG) (Sleator and Temperley, 1993) is also a dependency-based representation. This formalism is relatively similar to DepG, but it is enriched with forward or backward directions in the relations between the words, as represented in Figure 3.7. The properties that make the linkage possible are stored in a dictionary for each word. Moreover, the type of the relation is also determined for each lexical entry. As represented in Figure 3.7, 'D' determines a determiner relation, 'O' determines an object relation, and 'S' determines a subject relation. The syntactic analysis of Example 3.2a by LinkG is represented in Figure 3.8.

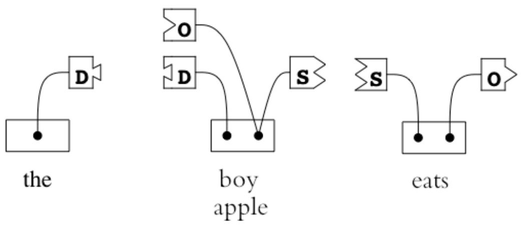


Figure 3.7: Lexical entries of Example 3.2a defined based on LinkG

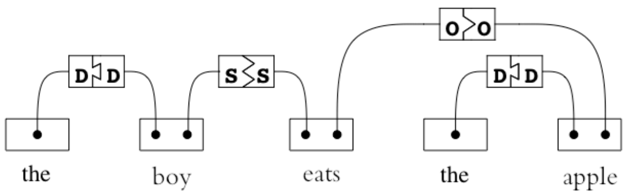


Figure 3.8: Syntactic analysis of Example 3.2a by LinkG

3.2.3 Head-driven Phrase Structure Grammar

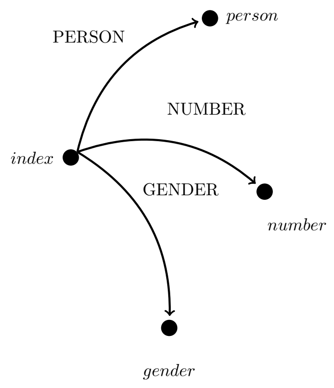
Head-driven Phrase Structure Grammar (HPSG) is a constraint-based linguistic theory that was first introduced in the mid-1980s by Pollard and Sag (1987). The applicability of the formalism for empirical problems are demonstrated in Pollard and Sag (1994).

Based on an assumption in HPSG, feature structures are considered as complete models of linguistic objects. Linguistic generalizations are expressed as constraints in this formalism. The constraints are descriptions of well-formed feature structures for a particular language. Constraints are formulae of a logical description language as feature value pairs which are usually represented in the form of AVMs. Each feature structure is required to be of a certain type. The types are ordered hierarchically in such a way that the most general type is at the top of the hierarchy and the most specific type at the bottom (Müller, To Appear, Sec. 1). Figure 3.9 represents the AVM of the type *index*.
In this AVM, the type *index* has the features PERSON, NUMBER, and GENDER. The value of the features can be atomic or partial. If a type has no attributes, it is called 'atomic'. The atomic features do not have sub-types. The feature description of a feature structure can be partial, i. e. maximally specific types are not used, or one or more attributes are omitted. The partial features

have sub-types. Feature descriptions can also be represented as a DAG or linear. The equivalent DAG and linear representation of the AVM in Figure 3.9 is shown in Figure 3.10.

$$\begin{bmatrix} \textsc{person} & \textit{person} \\ \textsc{number} & \textit{number} \\ \textsc{gender} & \textit{gender} \\ \textit{index} & \end{bmatrix}$$

Figure 3.9: AVM of the type *index*



index & PERSON(X1,X2) & *person*(X3) & NUMBER(X1,X4) & *number*(X5) & GENDER(X1,X6) & *gender*(X7)

Figure 3.10: DAG and linear representation of an AVM

Another property of this formalism is the inclusion of a small number of principles and grammar rules (schemas), and massive information in the lexical entries, which makes it highly lexicalized. In this formalism, the lexical knowledge, especially the lexical knowledge of heads, plays the most important role in providing the syntactic analyses. Additionally, each lexical entry is well-typed and knowledge is ordered hierarchically according to an ontology.

Structure sharing is another property of this formalism "that is the token identity between substructures of a given structure in accordance with lexical specifications or grammatical principles" (Pollard and Sag, 1994, p. 2). Structure sharing is indicated by numbered boxes in the feature descriptions.

HPSG is a constraint-based formalism, i. e. the feature descriptions of lexical items or phrases are the constraints, and only those feature structures are well-formed which satisfy all the constraints of the grammar.

HPSG is a system of *signs* which include *words* and *phrases*. Words carry

$$\left[\begin{array}{ll}
\text{PHON} & \langle \textit{the} \rangle \\
\text{SYNSEM} & \left[\begin{array}{ll}
\text{LOC} & \left[\begin{array}{ll}
\text{CATEGORY} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{ll} \text{SPEC} & \textit{noun} \\ \text{DEF} & \textit{plus} \\ \textit{det} & \end{array}\right] \\
\text{SUBJ} & \langle\rangle \\
\text{COMP} & \langle\rangle \\
\text{SPR} & \langle\rangle \\
\textit{category} &
\end{array}\right] \\
\text{CONTENT} & \left[\begin{array}{ll} \text{DET} & \textit{the} \\ \textit{content} & \end{array}\right] \\
\textit{loc} &
\end{array}\right] \\
\textit{synsem} &
\end{array}\right] \\
\textit{word} &
\end{array}\right]$$

Figure 3.11: Feature description of the word 'the' in Example 3.2a

two features: phonology (PHON) that encodes the phonological representation of the word, and syntax-semantics (SYNSEM) that encodes the syntax and semantic information of the word. Phrases carry three features such that in addition to the PHON and SYNSEM features, the feature daughters (DTRS) is used for constituent structures. The information in SYNSEM is represented in the features local (LOC) and non-local (NON-LOC). The LOC feature contains the information about the syntactic category (CATEGORY), semantic content (CONTENT), and discourse context (CONTEXT)[1]. The NON-LOC feature is used for unbounded dependencies, and it has two features, namely INHERITED and TO-BIND to represent the information about the extraposition of an element and binding the extraposed element. Either of the features INHERITED or TO-BIND contains the information about the extraposition of an element (SLASH), a WH-element (QUE), or a relative clause (REL).

Figures 3.11, 3.12, and 3.13 display the AVMs of the words '*the*', '*boy*', and '*eats*' in Example, 3.2a, respectively. As represented in the feature description of the word 'the', this definite determiner selects a noun through its SPEC feature. The feature description of the word '*boy*' states that this head noun requires a determiner as its specifier (SPR). The CONTENT | INDEX feature defines the morphological properties of this noun, which is a third person singular noun. The feature description of the word '*eats*' states that in the argument structure of this finite verb as the head, a noun phrase, which is third person singular in the nominative case, is required as its subject (SUBJ), and another noun phrase in the accusative case is required as its complement (COMP).

To put the words together to create an acceptable sentence, a limited number of schemas and principles are required. All of these schemas and principles

[1]The CONTEXT feature contains the information about background and pragmatics. This feature usually is not mentioned in AVMs.

$$\begin{bmatrix} \text{PHON} & \langle \mathit{boy} \rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CATEGORY} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CASE} & \mathit{nom} \\ \mathit{noun} & \end{bmatrix} \\ \text{SUBJ} & \langle\rangle \\ \text{COMP} & \langle\rangle \\ \text{SPR} & \langle \text{DET} \rangle \\ \mathit{category} & \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{NUM} & \mathit{sg} \\ \text{PERS} & \mathit{3rd} \\ \mathit{index} & \end{bmatrix} \\ \mathit{content} & \end{bmatrix} \\ \mathit{loc} & \end{bmatrix} \\ \mathit{synsem} & \end{bmatrix} \\ \mathit{word} & \end{bmatrix}$$

Figure 3.12: Feature description of the word ‘boy’ in Example 3.2a

$$\begin{bmatrix} \text{PHON} & \langle \mathit{eats} \rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CATEGORY} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{VFORM} & \mathit{fin} \\ \text{TENSE} & \mathit{pres} \\ \mathit{verb} & \end{bmatrix} \\ \text{SUBJ} & \left\langle \text{NP} \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CASE} & \mathit{nom} \end{bmatrix} \\ \text{INDEX} & \begin{bmatrix} \text{NUM} & \mathit{sg} \\ \text{PERS} & \mathit{3rd} \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{COMP} & \left\langle \text{NP} \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CASE} & \mathit{acc} \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{SPR} & \langle\rangle \\ \mathit{category} & \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \text{INDEX} & \begin{bmatrix} \text{NUM} & \mathit{sg} \\ \text{PERS} & \mathit{3rd} \\ \mathit{index} & \end{bmatrix} \\ \mathit{content} & \end{bmatrix} \\ \mathit{loc} & \end{bmatrix} \\ \mathit{synsem} & \end{bmatrix} \\ \mathit{word} & \end{bmatrix}$$

Figure 3.13: Feature description of the word ‘eats’ in Example 3.2a

also belong to the constraints in HPSG. The type and the requirements of the constituent structures are defined in the DTRS feature according to the ontology of phrases. In the following paragraph, the sketch of a number of principles and schemas in HPSG is described informally.

Most principles function as universals in HPSG like the Head-Feature Principle where “the HEAD value of any headed phrase is structure-shared with the HEAD value of the head daughter” (Pollard and Sag, 1994, p. 34). The Immediate Dominance (ID) Principle is the disjunction of the ID schemas among which only one schema should be used. The ID schemas function as grammar rules. The type of schema that should be selected depends on the type of the DTRS feature and the phrase ontology. The Head-Complement, Head-Subject, Head-Adjunct, Head-Specifier, Head-Marker, and Head-Filler schemata are among the basic ID schemata in HPSG. The Head-Complement-Schema is used for combining the head with the elements specified in the COMP feature to construct a phrase. The Head-Subject-Schema is used when all the elements in the COMP feature of the head verb are saturated. This schema realizes the subject of the sentence. The Head-Adjunct-Schema is used when the daughter element is not a head and this element is not selected as a complement or subject. The Head-Specifier-Schema is used for combining a specifier and a noun. The Head-Marker-Schema is used for phrases with complementizers recognized as markers. The Head-Filler-Schema is used for binding the slashed elements of unbounded dependencies. The NON-LOC | INHERITED | SLASH feature contains the information about an extraposed dependent in a phrase. The Head-Filler-Schema binds the slashed element in the NON-LOC | INHERITED | SLASH feature and makes it structure shared with the element in the NON-LOC | TO-BIND | SLASH feature.

In Example 3.2a, the determiner '*the*' and the noun '*boy*' are combined together through the Head-Specifier-Schema to construct a noun phrase as represented in the CATEGORY | HEAD | SPEC value of the determiner, and in the CATEGORY | SPR value of the noun in Figures 3.11 and 3.12. Likewise, the Head-Specifier-Schema licenses the combination of '*the*' and '*apple*' to construct a noun phrase. The Head-Complement-Schema is utilized to specify the requirements in the COMP feature of the head verb to construct a verb phrase. The Head-Subject-Schema is used then to saturate the requirements in the SUBJ feature of the verb to construct a verb phrase. The Head-Feature Principle ensures that the HEAD values of the mother node and the head daughter node are token identical. The effect of this principle is that the information located under the HEAD feature is projected from the head-daughter to the dominating phrase. Semantics can be expressed more formally by Minimal Recursion Semantics (MRS) (Copestake et al., 2006) in which it reformulates the constraints at lexical and phrasal levels from semantic perspective and allows for underspecification of scope constraints.

When all lexical requirements are satisfied and the constrains are specified, the analysis of the sentence is complete as represented in Figure 3.14.

3.2.4 Previous Studies on Persian HPSG

In the previous section, the basic concepts of HPSG were defined. In this section, we briefly review the HPSG literature for Persian.

Taghvaipour (2005) provided an HPSG analysis for relative clauses and free relative clauses as unbounded dependencies at the theoretical level in his Ph.D. dissertation. In this study, Taghvaipour divided the analyses of relative clauses

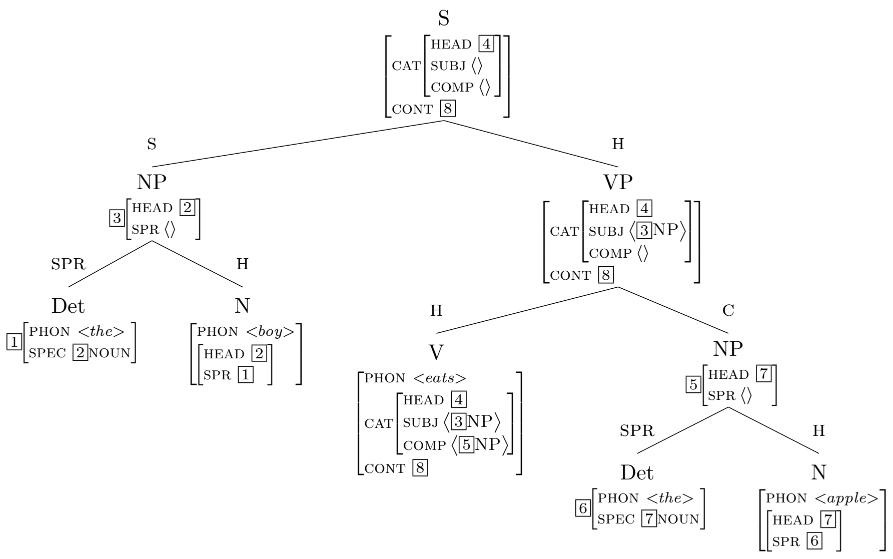


Figure 3.14: HPSG analysis of Example 3.2a

and free relative clauses into two groups and studied them in terms of the existence of a gap (trace) or a resumptive pronoun. The traceless and trace-based HPSG analyses are provided for both relative clause and free relative clause constructions. To provide the analyses, it was attempted to use a single mechanism. To this end, the SLASH and GAPTYPE features are defined in the NON-LOC attribute. The value of the SLASH feature is structure shared with the local dependency that is a head noun. The GAPTYPE feature can have either a *resumptive pronoun* value or a *trace* value.

Samvelian (2007) had a study on the analysis of Ezāfe. In this study, Ezāfe is treated as an affix attached to nouns, adjectives, and some prepositions, at both the lexical and phrasal levels. Ezāfe functions as a marker and it marks nouns to expect adjectives as their modifiers or nouns as their complements. Similarly, the post-nominal indefinite determiner 'ی' /-i/ and personal clitics, which behave like Ezāfe, are treated as phrasal affixes as well. The reason to treat them as phrasal affixes is that they occur at the left edge[2] of nominal phrases, and they are placed after the inflectional plural maker affix 'ها' /-hā/. The double functionality of Ezāfe is also discussed. Ezāfe functions as a word affix when it is added to a head, such as a noun, adjective, or preposition. But it functions as a phrasal affix when it is added to a modifier. The former functionality requires a lexical suffixation rule when the syntactic rules are applied; and the latter functionality is applied once when the syntactic rules have created a phrase through the lexical suffixation rule.

[2]The standard writing direction of Persian is right-to-left; and the left-hand side of the element (either word or phrase) is the outer edge.

Bonami and Samvelian (2009) studied five periphrastic constructions of verb conjugation, such as the auxiliary verb 'شدن' /šodan/ 'become' in passive voice, the verbal clitic and the auxiliary verb 'بودن' /budan/ 'be' in the present and past perfect tenses, the auxiliary 'خواستن' /xāstan/ 'will' in future tense, and the auxiliary 'داشتن' /dāštan/ 'have' in the progressive tense. They then proposed solutions to treat periphrastic constructions lexically rather than as phrasal or multi-word lexical item. In their study, lexical entries for each of these auxiliaries are provided. To this end, they integrated the Paradigm Function Morphology (Stump, 2001) in HPSG and adapted it for the target periphrastic constructions of Persian. In the analyses provided, these auxiliaries are licensed as heads.

Samvelian and Tseng (2010) had a study on clitic phenomena like pronominal clitics, possessive clitics, pre-verbal object clitics, clitic doubling constructions, and object clitics in the verbal domain. They presented a morpho-syntactic analyses for them. In their study, these clitics are treated as suffixes, and when they have wide scope (phrasal) they will function as a phrasal affix.

Müller (2010) studied light verb constructions. One salient feature of this study in light of the previous studies is that the proposed ideas are implemented within the TRALE system (Penn, 2004), an HPSG parser and generator, and the semantic analyses use MRS (Copestake et al., 2006). The MRSes are visualized by UTOOL (Koller and Thater, 2005). In this paper, first the author argues the shortcomings of inheritance hierarchies for classifying complex predicates proposed in Goldberg (2003). The author then enumerats several problems of Goldberg's analysis. The first problem is the semantic interpretation of a complex predicate combined with the future auxiliary, where the semantic interpretation of the complex predicate should be embedded under the meaning of the future auxiliary. The other problems address the interpretation of negation, the interaction between negation and object clitics and progressive/subjunctive marking, the interaction of negation and the future auxiliary and their semantic interpretation, the interaction between the future auxiliary, clitics, and complex predicates, and finally the separation of pre-verbal elements in complex predicates.

Müller and Ghayoomi (2010) performed a further study on Persian grammar and explained the various phenomena covered in the grammar along with the implementation of morphological rules, syntactic principles and schemas in the TRALE system (Penn, 2004), and semantic interpretation as MRSes by UTOOL. The implemented Persian grammar has a common core shared between several other languages (Müller, 2013), namely German (Müller and Ørsnes, 2011), Mandarin Chinese (Müller and Lipenkova, 2009), Danish (Ørsnes, 2009), Maltese (Müller, 2009), Yiddish, English, Spanish, and French. The implemented grammar for Persian uses a Head-Adjunct-Schema, a Head-Complement-Schema, a Head- Specifier-Schema, and a Head-Filler-Schema. In addition to these schemata, a Head-Cluster-Schema is used for complex predicates. In Müller and Ghayoomi (2010), it is explicitly defined how a lexical item belonging to various syntactic categories is defined in the grammar. It is further described that the lexical rules in the system cover both inflectional and derivation morphological rules, such as the plural maker affix, post-nominal determiner, the Ezāfe morpheme, and negation. In the lexicon, the lexical items, such as nouns, verbs, clitics, prepositions, direct object marker, determiners, and symmetric coordination are defined, and the number and the type of elements in their argument structures are determined. Additionally, dropness of the subject, empty categories, and the embedded clauses marked by the complementizer or the relativizer 'که' /ke/ 'that' are investigated. Finally, a small test suite is used for testing the implemented fragment of the grammar in the TRALE system (Penn, 2004) is evaluated. This test suite contains 165 sentences, and both positive and negative samples are involved. The grammar is able to parse all of the positive samples without providing any analyses for negative samples.

3.3 Treebanking

The term 'treebank' appears to have been coined by Geoffrey Leech (Sampson, 2003). A treebank is an annotated corpus constructed of a collection of sentences with their corresponding syntactic tree analyses. Therefore, the provided analyses are one level higher than assigning POS tags in which the syntactic category of each word in a sentence is determined. Generally, treebanking can be theory-independent like the TüBa-D/Z[3] developed for German, or it can be dependent on a linguistic theory. If the latter approach is chosen, then based on the grammar formalisms described in Section 3.2, various treebanks might be developed for a target language. In terms of the usage of treebanks, they can be used as language resources by linguists to investigate specific phenomena and test hypotheses in the target language, or by computational linguists who want to analyze sentences automatically for natural language understanding.

In the followings, first we present the previous studies and the methods of treebanking; and then we review the previous studies on Persian treebanking.

3.3.1 Previous Studies on Treebanking

Needless to say, theory-dependent treebanks require a grammar formalism as their backbone. Consequently, a number of treebanks might be constructed

[3]http://www.sfs.uni-tuebingen.de/en/ascl/resources/corpora/tueba-dz.html

based on either the syntactic constituent or dependency structure of sentences in each language. It seems that the pioneer research on treebanking is the development of Talbanken[4] in the 1970s at Lunds University for Swedish that contains written and spoken Swedish (Teleman, 1974). The treebank contains POS tags, and implicit head-dependent relations, which are defined hierarchically in the constituent structures. The development of the Prague Dependency Treebank is among the earliest attempts to develop a data source with dependency relations for Czech (Hajič, 1998; Böhmová et al., 2003), and it is further on developed for other languages like English (Hajič et al., 2012).

In the followings, we describe the basic properties of the well-known treebanks that are developed semi-automatically for the English, German, and Bulgarian languages, as well as the treebanks developed through an automatic conversion.

One of the first large-scale treebanks is the Penn Treebank for English developed by Marcus et al. (1993). This treebanking is done in the framework of Chomsky's PSG (Chomsky, 1957) with relatively flat syntactic constructions. The annotated texts in the treebank include a wide-range of genres, such as IBM computer manuals, nursing notes, Wall Street Journal articles, and transcribed telephone conversations (Taylor et al., 2003). Out of approximately 7 million words in the text corpus that are POS tagged, 3 million words are parsed, and over 2 million words of the corpus are parsed for predicate-argument structure. The first release of the treebank contains limited empty categories without the indication of non-contiguous structures and dependencies. In the revised version, the Penn Treebank II, traces are co-indexed with the lexical elements. Traces are inserted to the trees to indicate long-distance dependencies. Moreover, the treebank is enriched with a small number of syntactic functions including numerical indices for explicitly marking the logical subject and logical objects of the verbs, and defining how sub-constituents are semantically related to their predicates. In this data set, the type of the relation between the elements of a constituent as either a complement or an adjunct is not explicitly determined for all constituents. POS tagging and syntactic parsing of the texts are the product of a two-step method: automatic annotation followed by manual correction. For POS tagging, the stochastic PARTS POS tagger (Church, 1988) is trained with a modified version of the Brown Corpus, and the output is mapped to the Penn Treebank tag set and corrected manually. Next, this data is used for training the deterministic Fidditch parser (Hindle, 1983). The parse results are finalized by human intervention and manual correction using a task-specific mouse-based interface implemented in GNU Emacs Lisp.

The TIGER Treebank (Brants et al., 2002) is a treebank for German de-

[4]`http://w3.msi.vxu.se/~nivre/research/talbanken.html`

veloped in the framework of Chomsky's PSG (Chomsky, 1957). The syntactic structures in the treebank are enriched with a set of syntactic functions to determine the type of relations either as an argument or adjunct. The tree structures in this treebank are relatively flat. The early version of this treebank contains 35,000 sentences from newspaper texts which aimed at being extended to 80,000 sentences. The treebank's annotation scheme is an extended version of the NEGRA Corpus annotation scheme (Skut et al., 1997, 1998). This treebank is developed in two steps: interactive annotation, and LFG parsing. After POS tagging the data automatically with the TnT POS tagger (Brants, 2000), the Cascaded Markov Model (Brants, 1999a,b) trained on the NEGRA Corpus is used for generating phrase-by-phrase annotation in which a human annotator can intervene immediately after each new proposed phrase. Based on the annotator's decision, the parser proposes the next part of the annotation. This process is repeated until the annotation of the sentence is complete. In this method, both the tagger and the parser are implemented within the Annotate tool (Plaehn and Brants, 2000), and no manual grammar rules or lexicon development is necessary. In the second step, parts of the corpus are parsed with the LFG parser developed for German (Dipper, 2000), and the output is disambiguated by a human annotator. A transfer module (Zinsmeister et al., 2001) is employed to convert the selected LFG analysis to the TIGER format, which is the flat phrase structure tree enriched with syntactic functions. This module is also used for adding the tree to the treebank. The developed treebank is enhanced with a search engine for queries (Lezius, 2002) and a query language (König and Lezius, 2002a,b).

The Linguistic Grammars Online (LinGO) Redwoods Treebank is an HPSG treebank for English. This treebank is a collection of manually annotated corpora analyzed with the LinGO English Resource Grammar (ERG) which contains complete syntactic and semantic analyses (Flickinger, 2000). This treebank is developed through a dynamic approach. One direction of this dynamic approach is dealt with the ways to retrieve the linguistic data from the treebank during varying the granularity level in the treebank. The other direction of this dynamic approach is updating the treebank regularly synchronized to the development of ideas in the syntactic theory (Oepen et al., 2004). Several tools are used for developing this treebank, and it is possible to extract different types of linguistic information from the treebank. The treebank is built on the [incr tsdb()] profiling environment (Oepen and Callmeier, 2000; Oepen, 2001). The tree comparison tool in the Linguistic Knowledge Building (LKB) platform (Copestake, 2002) is used for extracting the elementary linguistic properties (the discriminants). Carter's inference rules (Carter, 1997) are used for determining the smallest set of discriminants. This disambiguated set is presented to the annotators to navigate through the proposed parse trees and make a binary decision on selecting either the correct analysis or the most preferred analysis of the sentence with respect to the context, or rejecting it. In the dynamic treebanking, the annotator's choices are inferred automatically from the recorded disambiguating decisions in the [incr tsdb()] database to re-apply the disambiguating decisions of the updated grammar on the corpus. Three different forms of information are available in the [incr tsdb()] environment: (a) a derivation tree for representing the full HPSG analysis, (b) a normal phrase structure tree, and (c) MRSes (Copestake et al., 2006) for semantic representation which look like elementary dependency graphs extracted from MRSes. The nodes of the dependency graphs are the MRS relations, and the arcs are labeled by MRS roles. 10,000 sentences of the transcribed dialogues from the Verbmobil data (Wahlster, 2000) are used as initial data. The PET unification-based parser (Callmeier, 2000) is used for providing the HPSG analysis.

The DeepBank is another HPSG treebank for English (Flickinger et al., 2012). This treebank has adopted the dynamic approach described above, and it is attempted to overcome the shortcomings in manual annotation of a corpus. The dynamic approach helps to refine the treebank by the improvement of the grammar. In the development of this treebank, it is aimed at providing the syntactic and semantic analyses of Wall Street Journal articles in the English Penn Treebank. The tools that are used for this treebanking are the PET parser (Callmeier, 2000), the LKB platform (Copestake, 2002) for grammar development, the [incr tsdb()] profiling environment (Oepen and Callmeier, 2000; Oepen, 2001), and the ERG (Flickinger, 2000). Although the data set which is used for this treebanking has already been annotated, the input data is transformed into raw data, which is then fully re-annotated independent of its original annotation. In the annotation process, each section of the Wall Street Journal corpus is parsed with the PET parser trained with the ERG. Meanwhile, the TnT POS tagger (Brants, 2000) is employed to assign a POS tag to unknown words. Next, the n-best (n=500) parse trees of each sentence are extracted and ranked using a maximum entropy model built with the TADM package (Malouf et al., 2000). The parse trees are stored in the [incr tsdb()] treebanking tool, and the derivation tree of each parse tree is recorded in this tool. Human annotators disambiguate the parse result of each sentence manually in the [incr tsdb()] treebanking tool with a binary decision, either selection or rejection.

The Bulgarian TreeBank (BulTreeBank) is an HPSG-based treebank for Bulgarian (Simov et al., 2002b). The development of this treebank is started from scratch. To start data annotation, two sets of data are provided from two sources: examples from the Bulgarian grammar books for basic linguistic phenomena (1,500 sentence), and free texts from newspapers, government documents, and proses (10,000 sentences) (Osenova and Simov, 2004). King's Speciate Re-entrance Logic (SRL) (King, 1989) is used for representing an HPSG grammar, therefore feature graphs as finite signatures are defined for representing the HPSG sort hierarchy and principles within the CLaRK[5] system for developing the treebank. The system as well as its properties are described in more detail in Section 5.2.

Conversion of treebanks from one grammar formalism to another, or creating a parallel treebank is another strategy for treebanking. This conversion can be done automatically or manually.

In the former model of conversion, a monolingual treebank based on one grammar formalism is converted into another formalism, for instance converting a treebank from LTAG into HPSG for English (Tateisi et al., 1998; Yoshinaga and Miyao, 2002), PSG into HPSG for English (Miyao et al., 2005), PSG into HPSG for German (Cramer and Zhang, 2010), PSG into HPSG for Chinese (Yu et al., 2010), DepG into HPSG for Russian (Avgustinova and Zhang, 2010), HPSG into DepG for Bulgarian (Chanev et al., 2006), PSG into CCG for English (Hockenmaier, 2003; Hockenamier and Steedman, 2007), DepG into CCG for Italian (Bos et al., 2009) and Turkish (Çakici, 2005), PSG into TAG for English (Chen et al., 2006b), PSG into LFG for English (Cahill et al., 2002; Burke, 2006) and Chinese (Guo et al., 2007) and Arabic (Tounsi et al., 2009) and French (Schluter, 2011), and PSG into DepG for French (Candito et al., 2010).

In the latter model of conversion, a parallel corpus is employed for creating a parallel treebank such that first a parallel corpus is parsed for each language individually, and then the parsed parallel corpus is automatically aligned at either sub-sentential or tree level, such as the parallel treebanks of PSG for Korean–English (Han et al., 2002), Estonian–German (Uibo et al., 2005), English–Spanish–German (Tinsley et al., 2007), and English–French (Zhechev and Way, 2008), or the parallel treebank of DepG for Czech–English (Čmejrek et al., 2004). Samuelsson and Volk (2006) created a parallel treebank manually aligned between three languages, namely German, English, and Swedish.

3.3.2 Previous Studies on Persian Treebanking

There have been attempts to develop treebanks for Persian which are briefly described in this section.

The study of Pouramini and Mozayani (2007) is one of the first studies for developing a Persian treebank. In their study, they mainly focused on the

[5]`http://bultreebank.org/clark/`

methodological principles and the syntactic criteria to design an annotation scheme for developing a treebank for Persian without a dependency on any grammar formalism. After studying the pros and cons of PSG and DepG for Persian, they established a hybrid framework which combines the advantages of the both formalisms to annotate a Persian text. Their proposed approach is constructed based on the properties of two treebanks: (a) the NEGRA Treebank (Brants, 1997) for German, and (b) the Italian treebank (Montemagni et al., 2003). To this end, they introduced feature descriptions for the lexical items in their treebank. The feature descriptions contain three components: (a) Morpho-Syntactic-Component for lexical information, (b) Functional-Syntactic-Component for syntactic information, and (c) Semantic-Component for semantic information. Since they found similarities between their proposed method and the NEGRA annotation scheme, they used an incremental approach similar to the NEGRA project for Persian such that a human annotator interacts with a parser to either accept or reject the proposed structures. In this study, they used the data from FLDB (Assi, 1997) and the semi-automatic POS tagger developed by Assi and HajiAbdolhosseini (2000).
Although the study of Pouramini and Mozayani established the preliminary steps towards the development of a treebank for Persian, the paper lacks detailed information of the annotation process, as well as the amount of the annotated data. Additionally, neither qualitative nor quantitative evaluation has been done on the annotated data, and the amount of human intervention is not clear either.

SharifiAtashgah (2009) proposed a semi-automatic approach in his Ph.D. dissertation to generate a treebank for Persian in the framework of the Minimalist Program (Chomsky, 2000, 2001). To this end, an algorithm with a bottom-up approach was proposed for semi-automatic bracketing. The major problems that SharifiAtashgah has to dealt with are the tokenization and annotation of multi-tokens. In this study, the multi-tokens are divided into static and dynamic units. The static units are closed, unproductive units which can be listed, such as compound words, while the dynamic units are open and productive, such as the light verbs. The annotation scheme of the English Penn Treebank (Marcus et al., 1993) and its modification for annotating Persian text in the framework of the Minimalist Program are described in the dissertation as well. The defined constituent labels are: the Declarative Sentence (vP), Time Phrase (TP), Focus Phrase (FocP), Topic Phrase (TopP), Complement Phrase (CP), and Fragment Phrase (FRAG). Moreover, a set of syntactic labels, such as subject (−SBJ), object (−OBJ), predicate (−PRD), logical subject (−LGS), and vocative (−VOC), are added to the constituents. Additionally, a number of semantic labels, such as property (−PRP), time expression (−TMP),

direction (−DIR), location (−LOC), and manner (-MNR), are added to the constituent labels as well. In the annotation scheme, several empty categories, including trace (*T*), null complementizer or relativizer (0), pro-dropped NP (PRO (NP*)), canonical position of extracted constituent which is co-index with its reference (*ICH*[6]), and ellipsis (*PNR*), are defined.
To generate the parse trees of Persian sentences, an algorithm is introduced, whose architecture is demonstrated in Figure 3.15. The input data to the system is the POS tagged Peykare corpus (Bijankhan et al., 2011). First, the static and dynamic multi tokens are tokenized, then they are parsed. Finally, the shallowly processed sentences are sent to the syntactic component where the sentences are syntactically bracketed according to the defined grammar rules.
Even though a complete algorithm is proposed in the dissertation, the implementation is partial, and it is only limited to the recognition of noun phrases. Therefore, the current version of the system cannot parse a complete sentence. To evaluate the accuracy of the system for recognizing the noun phrases, two text files which contain 6,273 word tokens are randomly selected and given to the system for bracketing. Based on the given data sets, 3,452 noun phrases are bracketed out of which 248 brackets are wrong; i. e. the error rate of the system for bracketing noun phrases is 7%.

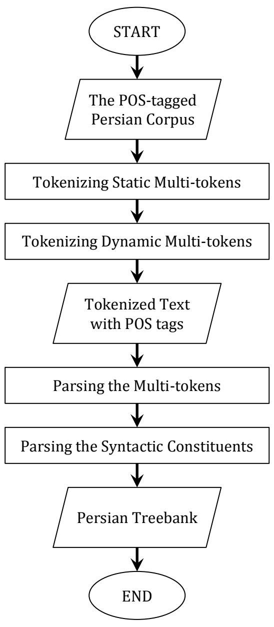

START
The POS-tagged Persian Corpus
Tokenizing Static Multi-tokens
Tokenizing Dynamic Multi-tokens
Tokenized Text with POS tags
Parsing the Multi-tokens
Parsing the Syntactic Constituents
Persian Treebank
END

Figure 3.15: Architecture of the system proposed by SharifiAtashgah (2009)

[6]ICH stands for ‘Interpret the Constituent Here’. This element functions as the trace.

Pouramini and Moridi (2012) only described an annotation scheme to be used for grammatical functions and extrapositions of relative clauses. Their general proposed annotation scheme is similar to Pouramini and Mozayani (2007), and it is enriched with additional features to handle grammatical functions and extrapositions of relative clauses. Again, their study is not evaluated either qualitatively or quantitatively.

Seraji et al. (2012a) recently described a method to develop a dependency treebank for Persian, called Uppsala Persian Dependency Treebank (UPDT)[7], through a bootstrapping approach. This data set is released online. A set of 215 manually annotated sentences is used as the seed data to initialize and train the Malt parser (Nivre et al., 2006), the non-probabilistic, statistical parser for DepG. Since this parser is language independent, it can be trained with the data of any language to create the DepG model. Iteratively a number of sentences are annotated and added to this treebank. As reported, the primary version of the treebank is iteratively increasing up to 1,000 sentences. Seraji et al. (2012c) has stated that the aim is to enlarge the size of the treebank to 10,000 sentences from the Bijankhan Corpus (Bijankhan, 2004). This fragment of annotated data includes both long and short sentences, and the average sentence length in this data set is 19 words. The annotation scheme used in the study is mainly based on the Stanford Typed Dependencies (de Marneffe and Manning, 2008) along with four newly defined dependency relations, such as *acc* as accusative marker for direct objects, *ez* for Ezāfe constructions, *int* for interjections, and *lvs* for light verb constructions (*acomp-lvc* for adjectival complement in Light Verb Construction (LVC), *dobj-lvc* for direct object in LVC, *nsubj-lvc* for nominal subject in LVC, and *prep-lvc* for prepositional modifier in LVC). Training the Malt parser with the initial developed data set and evaluating it based on the 10-fold cross validation resulted in 56.7% accuracy for labeled attachment[8]. It is further reported that the extended version of the data set (1,000 sentences) is used for training two dependency parsers, namely the Malt parser and the Maximum Spanning Tree (MST) parser (McDonald et al., 2005). The experimental results of using gold POS tags have shown that the performance of the Malt parser is increased to 68.68% for labeled attachment and 74.81% for unlabeled attachment[9] while the accuracy of the MST parser is 63.60% for labeled attachment and 71.08% for unlabeled attachment.

The Persian Dependency Treebank (PDT) is also a recent dependency treebank for Persian developed through a bootstrapping approach (Rasooli et al.,

[7]`http://stp.lingfil.uu.se/~mojgan/UPDT.html`

[8]Labeled attachment is a relation with the correct head and the correct dependency label.

[9]Unlabeled attachment is a relation with the correct head disregarding the dependency label.

2013) in the Dādegān group[10]. This data set is also available online. This data set is developed according to the Tenth Conference on Computational Natural Language Learning (CoNLL-X) shared task on multi-lingual dependency parsing (Nivre et al., 2007). In this study, the MST parser (McDonald et al., 2005) is used. This treebank, which contains 29,982 sentences (498,081 word tokens) with an average length of 16.61 words, is constructed from a corpus composed of the online archives of the Mehr News Agency (3,000 sentence), art magazines, randomly selected sentences from the contemporary Persian literary texts of famous writers, articles, transcribed speeches of famous speakers, and the teaching texts used for teaching Persian to non-native speakers. Their annotation depends heavily on the valence of nouns, adjectives, and verbs. To this end, a Persian verb valence dictionary (Rasooli et al., 2011) is developed based on the Bijankhan Corpus (Bijankhan, 2004) and the Hamshahri Corpus (AleAhmad et al., 2009). The performance of the parser using raw text is reported to be approximately 75% for labeled attachment and 80% for unlabeled attachment.

The HPSG (Pollard and Sag, 1994) properties and also the available recent studies on Persian HPSG at both the theoretical level (Taghvaipour, 2005; Samvelian, 2007; Bonami and Samvelian, 2009; Samvelian and Tseng, 2010; Müller, 2010) and the system implementation level (Müller, 2010; Müller and Ghayoomi, 2010) motivated us to choose this formalism as the backbone of our treebank. Due to the lack of any publicly available syntactically annotated data when the present research commenced in 2009, either in the form of a constituency or a dependency treebank, we decided to develop a treebank for Persian and to release it online for free. Consequently, we started the annotation task from scratch. To this end, we use a bootstrapping approach for treebanking that is described in detail in Chapter 5. To maintain consistency in the analyses throughout the treebank, we need an annotation scheme that is described in Chapter 4.

3.4 Parsing

Allen (1995, p. 41) defined parsing "as a method of analyzing a sentence [automatically] to determine its structure according to [a] grammar". In Section 3.2, we briefly introduced a number of well-known grammar formalisms. Each formalism requires its own parsing method. But all parsing methods have properties in common. In this section, we briefly describe the basic and general properties of constituency- and dependency-based parsings.

[10]`http://dadegan.ir/`

3.4.1 Constituency-based Parsing

Constituents are the basic components of a sentence which are grouped together according to a topological grammar to construct the sentence. There are three approaches to develop the parsers: the rule-based, the statistical, are the hybrid approach. In the following paragraphs, the main properties of the rule-based and statistical approaches are briefly described. The description is limited to the basic ideas originated in constituency-based parsing according to PSG to analyze sentences automatically.

In the rule-based approach, a set of non-redundant competence grammar rules are written manually for the parser. Therefore, these grammar rules are the primary linguistic knowledge that the parser uses for analyzing a sentence. The parser takes a POS tagged sentence as the input, and it returns a labeled bracketed sentence as its output. The disadvantage of rule-based parsers is their dependency on a language; i. e. if the set of grammar rules is used for analyzing the sentences of one language, the defined grammar rules might not be applicable to another language. The advantage of rule-based parsers is that they do not require any annotated data at all to learn the linguistic knowledge from the data, but they require a set of grammar rules written manually by a grammarian.

In the statistical approach, a treebank is given to a statistical parser as the input training data. Next, the parser learns the linguistic knowledge from the data (supervised grammar induction), and it creates a grammar model based on the data. Consequently, the grammar rules are induced from the treebank automatically, and they are stored in the parser. The statistical information is extracted from the data as well to be used for building a probabilistic grammar model and to estimate the probabilities of the derivation trees of unseen data. A statistical parser produces multiple derivation trees for each sentence by using the equation (3.1) to estimate the probability of a tree t based on the derived trees d. To return a potential tree analysis of a sentence from the derived trees as the canonical derivation, the parser computes the probability of a derivation tree by using a chain of grammar rules r_i that are conditioned to the previous grammar rules (the history) from the collection of grammar rules as represented in the equation (3.2) (Manning and Schütze, 1999, pp. 422–423).

$$P(t) = \sum_{|d:\ d\ is\ a\ derivation\ of\ t|} \prod P(d) \tag{3.1}$$

$$P(d) = \prod_{i=1}^{m} P(r_i|r_1, ..., r_{i-1}) \tag{3.2}$$

Based on the equation (3.1), it is possible to have more than one derivation

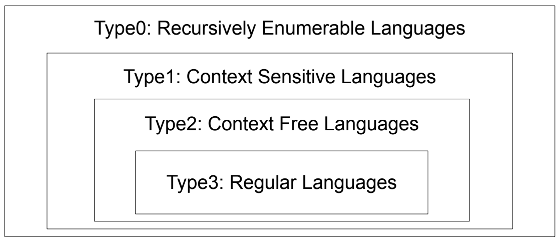

Type0: Recursively Enumerable Languages
Type1: Context Sensitive Languages
Type2: Context Free Languages
Type3: Regular Languages

Figure 3.16: Languages in Chomsky's hierarchy

parse tree for a sentence. In the case that each parse tree has one unique canonical derivation, the Viterbi algorithm (Viterbi, 1967) is used for finding the candidate derivation that has the highest probability score from the search space. "[T]he Viterbi algorithm is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise" (Forney, 1973).

What was briefly described above ideally represents the general approach used in statistical parsings. The disadvantage of such parsers is that they require a large amount of annotated data which has already been analyzed by experts. The development of this data is a labor intensive task. But the advantage of such parsers is that they are language independent, and they can automatically adapt to new domains, genres, or languages for inducing the linguistic properties and building a grammar model.

PSG described in Section 3.2.1 is a grammar formalism in linguistics. In computational linguistics, this formalism is called Context Free Grammar (CFG) according to Chomsky hierarchy. This hierarchy, displayed in Figure 3.16, is the hierarchy of four types of formal grammars introduced by Chomsky (1959). Type2 of the formal grammars is CFG. CFG is powerful enough to describe a great fraction of structures in a natural language. CFG additionally makes it possible to build efficient parsers for analyzing sentences (Allen, 1995, p. 42). Using probabilities for estimating CFG rules results in having a probabilistic grammar, called Probabilistic Context Free Grammar (PCFG).

Depending on the methodology used for modeling the grammar of a language, various statistical parsers are created. If the parser uses a lexical item as the head of a constituent, and the head is projected to the mother node, it is called a 'lexicalized parser'. If no knowledge of heads is used, the parser is called an 'unlexicalized parser'. The Stanford parser (Klein and Manning, 2003) is an implementation of the lexicalized parsing model, and the Charniak parser (Charniak, 1996) and the Berkeley parser (Petrov et al., 2006) are samples of the unlexicalized parsing model.

It should be pointed out that the general approach of statistical parsing that was described above uses probabilities to create a grammar model. However, it is possible to have a statistical parser that does not use probabilities. In our research, when we call a parser a 'statistical parser', we are referring to a supervised statistical parser that uses probabilities to build the grammar model. If the statistical parser only uses statistics and no probabilities to create a grammar model, we explicitly express that the parser does not use probabilities, such as the Link Grammar Parser (Sleator and Temperley, 1993) which is a non-probabilistic, statistical parser for LinkG, the Malt parser (Nivre et al., 2006) and the Mate parser (Bohnet, 2009) which are non-probabilistic, statistical parsers for DepG, and SXLFG (Boullier and Sagot, 2005) which is a non-probabilistic, statistical parser for LFG.

The output of a parser is a bag of trees. There exist two parsing techniques for searching the subtrees: (a) top-down, and (b) bottom-up (Ingerman, 1966). In the implementation of these parsing techniques, the derived partial structures are stored and organized in a chart, therefore the parsing process is called 'chart parsing'. In the following section, these two basic parsing techniques and their corresponding algorithms are described.

Top-down

In top-down parsing (Allen, 1995; Aho et al., 2007), the parser searches for a candidate parse tree from the bag of trees, and it builds the structure of the sentence from the root node S towards the leaves, i. e. the type of searching is goal-directed. In this parsing technique, the parser finds the top nodes of the trees which are started with the symbol S, and it derives the potential analyses based on the right-hand side of the grammar rules which provide new expectations for new grammar rules recursively. Next, with respect to the grammar rules that have the same left non-terminal labels, partial structures are derived. This process continues until it reaches the leaf nodes which are the words with their corresponding POS tags. At the end of the process, the trees which fail to match the words are eliminated. The result of this parsing technique is a set of potential parse trees of the target sentence. The advantage of this parsing technique is that the analyses which do not lead to the symbol S do not present in the chart parse. This technique, however, needs a huge amount of effort on the trees that contain the symbol S but do not match the input and they should be eliminated afterwards. Furthermore, the trees are generated without examining the input.

The Early algorithm (Earley, 1968) is an example of top-down parsing such that its chart has $(N + 1) \times (N + 1)$ cells where N is the number of words in the sentence. For efficient top-down search, a dynamic programming approach is used in this algorithm. In the dynamic programming approach, solutions are built from sub-solutions compositionally, and the sub-solutions are stored for reusing. The usage of the chart in this algorithm is for controlling the progress of the completed subtrees and determining the position of the subtrees. This algorithm involves four steps: start, prediction, scanning, and completion. First, the start symbols S should be searched and matched. Next, new expected states are created in the prediction step of the parsing process. In the scanning step, the input grammar rules are checked, and they are added to the chart with respect to their predicted POS tags while creating a new state from the input state. In the completion step, the right-hand side of grammar rules are covered completely and the left-hand side non-terminals are completed.

Bottom-up

Bottom-up parsing (Allen, 1995; Aho et al., 2007) is quite in contrast to top-down parsing. This parsing technique uses a data-directed search method to find structures in the bag of trees. A dynamic programming approach is also employed for this parsing technique in a chart. This parsing technique involves three steps: start, scanning, prediction. Contrary to top down parsing, there is no prediction for the expected cells. In this parsing technique, the parser takes the words of a sentence as the input and the tree is built from the words towards the top node S which dominates all of the input. When the right-hand side of grammar rules are matched with the partial analyses from the bag of trees, they produce the left-hand side non-terminals. If the right-hand side of grammar rules do not match, the parsing process stops. This process continues until it reaches the top node S. This parsing technique results in a set of potential parses for a given sentence. The advantage of this parsing technique is that it does not produce trees that potentially must not be produced due to the words and their corresponding POS tags. Using this parsing technique, however, does not guarantee that all derived trees lead to the node S.

Cocke-Kasami-Younger (CKY) algorithm (Kasami, 1965; Younger, 1967; Cocke, 1969) is an example of bottom-up parsing which is widely used for CFG parsing.

3.4.2 Dependency-based Parsing

In the previous section, the rule-based and statistical approaches for constituency-based parsing were described. Similarly, there are the rule-based, the statistical, and the hybrid approaches for dependency-based parsing.

The dependency parsing model proposed by Hays (1964) is among the first attempts to develop a rule-based parser based on DepG. In the proposed approach, the rules are defined based on the subcategorization requirements (argument structures) of the words in the lexicon. Grammar Rules 3.6 are defined according to DepG rules of Hays (1964). As demonstrated, the symbols in front of the brackets are the heads, and the symbols within the brackets are the dependents. The linear position of the head among its dependents is marked with an asterisk (*). The assigned POS tags to the words in the lexicon serve as the categories in the rules. Categories that are the heads in a sentence can serve as the root of the dependency tree, and they must be marked during the generation process. We use underline to determine the head in Grammar Rules 3.6.

(3.6)	$\underline{\text{V}}$ (N * N)	V = {studies, eats, etc}
	N (T *)	N = {boy, book, apple, etc}
	T (*)	Det = {the, a, an}

RelEx[11] is a semantic dependency relationship extractor for English that generates dependency trees. A series of graph rewriting rules are employed to identify subject, object, indirect object and many other syntactic dependency relationships between the words in a sentence.

In the statistical approach of dependency-based parsing, similar to statistical constituency-based parsing, training data is used for inducing the grammar model. The Malt parser (Nivre et al., 2006) and the Mate parser (Bohnet, 2009) are two non-probabilistic, statistical parsers that are described in Section 6.3.

In the hybrid approach, both the rule-based and the statistical approaches are used in the parsing algorithm. An advantage of this approach is that the grammar model is created from different methods, rather than to create the model from a treebank. The WCDG parser (Foth and Menzel, 2006) is an example of a hybrid dependency-based parsing for German. Sennrich et al. (2009) also developed a hybrid dependency-based parsing for German that uses a hand-written grammar with a probabilistic disambiguation system.
The hybrid approach cannot be built only based on the rule-based and statistical approaches, but the hybrid approach can be built based on the constituency- and dependency-based parsings, such as the parser developed by Hall et al. (2007) for Swedish.

3.4.3 Parsing Evaluation

Black et al. (1991), Goodman (1996), Carroll et al. (1998), and Manning and Schütze (1999) presented a wide variety of evaluation metrics to evaluate the performance of a parser. Among them, 'Precision', 'Recall', 'F-measure', 'Crossing Bracket rate', and 'Exact Matching rate' are the very well-known measures

[11] `http://wiki.opencog.org/w/RelEx_Dependency_Relationship_Extractor`

which are widely used. The three metrics, namely precision, recall, and F-measure, are together called the 'PARSEVAL metric'. The PARSEVAL metric evaluates each piece of information in a parse tree. The original PARSEVAL metric ignores the labels of the nodes and unary branching. But it is possible to include labels, and to have labeled precision and labeled recall. Evalb[12] is the implementation of the labeled PARSEVAL metric which computes labeled precision and labeled recall to evaluate both bracketing and labeling of the constituents.

Precision (P) and Recall (R) metrics measure the ratio of the number of correct constituents against the number of constituents parsed by the parser and the number of constituents in the gold data respectively. The gold data is the reference and standard data created by experts. This data is used for comparing the output of a parser with it to evaluate the performance of the parser. The equations (3.3) and (3.4) represent how these metrics are computed.

$$P = \frac{Number\ of\ Correct\ Constituents}{Total\ Number\ of\ Constituents\ in\ Parser\ Output} \tag{3.3}$$

$$R = \frac{Number\ of\ Correct\ Constituents}{Total\ Number\ of\ Constituents\ in\ Gold\ Standard} \tag{3.4}$$

F-measure is a weighted harmonic mean of precision and recall. This metric is recognized as a standard evaluation metric for the accuracy of constituency parsing represented in the equation (3.5).

$$F - measure = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)\times P \times R}{\beta^2 \times (P+R)} \tag{3.5}$$

where $\alpha = \frac{1}{\beta^2+1}$, $\alpha \in [0,1]$, $\beta \in [0,+\infty]$, P is precision, and R is recall. Assigning a small value to β ($\beta < 1$) gives higher weight to precision, and assigning a high value to β ($\beta > 1$) gives higher weight to recall. Assigning the value 1 to β ($\beta = 1$), known as F_1, means that both precision and recall have the same priority as represented in the equation (3.6).

$$F - measure = \frac{2 \times P \times R}{P+R} \tag{3.6}$$

The Crossing Bracket (CB) rate measures the average bracketing of the parser's output which is produced, say (A (B C)), against the number of gold data which provides brackets like ((A B) C). The crossing bracket rate is evaluated according to the equation (3.7).

$$CB = \frac{Num.\ of\ Cons.s\ in\ Parser\ Output\ that\ Cross\ Gold\ Cons.s}{Total\ Number\ of\ Constituents\ in\ Parser\ Output} \tag{3.7}$$

[12]`http://nlp.cs.nyu.edu/evalb/`

The Exact Matching (EM) rate, also called ‘tree accuracy’, is a tough criterion which assigns the score 1 to a completely correct parse tree and 0 if there is any kind of mistake in the parse tree as represented in the equation (3.8).

$$EM = \frac{Number\ of\ Correctly\ Analyzed\ Sentences}{Total\ Number\ of\ Analyzed\ Sentences} \tag{3.8}$$

Carroll et al. (1998) pointed out a shortcoming of the PARSEVAL metric. If the bracketing and labeling are successful, then a complete score is assigned to the structure; but in the case that there is any failure, no score is assigned. Sampson (2000) proposed the Leaf-Ancestor (LA) metric which is a string-based measure using the Levenshtein distance (Levenshtein, 1966), and it computes a cost for converting a false label to the correct one. This metric compares the similarity of the path to link each leaf (word) of a sentence to the root node in both of the gold tree and the candidate tree, and it computes the overall average of the correct paths. In this evaluation metric, if there is a mismatch on one constituent, only the score of the words in that constituent is affected and this failure is reflected in the overall average. This metric “quantifies the human perceptions of relatively accurate or inaccurate parsing” (Sampson, 2000).

A disadvantage of LA is its sensitivity to the number of brackets in the trees. Lin (1998) proposed another evaluation strategy in which the constituent structures are converted to dependencies, and then the dependency relations of the elements with the heads rather than the constituent boundaries are measured. In this evaluation metric, each single word is involved in the evaluation, and it is easier to search for errors. Labeled and unlabeled attachments are two standard metrics for evaluating the dependency relations. The labeled attachment metric measures the ratio of relations with the correct head and the correct dependency label to the total number of relations. The unlabeled attachment metric measures the ratio of relations with the correct head and disregarding the dependency label to the total number of relations.

3.4.4 Previous Studies on Parsing Persian

The lack of a treebank as a data resource for the Persian grammar has made it impossible to train a statistical parse for Persia. This has restricted researchers to analyze the sentences at the chunk level with a limited tree depth rather than the hierarchical constituents. As another alternative, they have used the rule-based, semi-supervised, or unsupervised parsers.

Kiani et al. (2009) used a hybrid model for chunking Persian. In this approach, first a rule-based chunker is used for labeling Inside/Outside/Beginning segments. Next, a multilayer perceptron neural network is exploited for training the system with the provided chunks. On the top of the neural network, a fuzzy C-means clustering method (Bezdek et al., 1984) is used for chunking new sentences.

To the best of our knowledge, the study of Sanamrad and Matsumoto (1986) is among the first studies to develop a parser for Persian. Their developed parser, called PERSIS (PERsian analySIS), is a rule-based parser written in Lisp to bracket an input sentence and to provide a dependency network for representing meaning. The dependency network they provided is similar to the conceptual dependency introduced by Schank (1975). RaisGhasem (1991) also developed a parser to produce the conceptual dependencies.
Rezaei (1993) built an Augmented Transition Network (ATN) system for parsing Persian. This system is a constraint- and rule-based parser that analyzes a substantial fragment of Persian. In the extended version of the ATN system, the author used a two-step constraint-based parsing written in Prolog to handle scrambling of the constituents in languages with free constituent order like Persian (Rezaei, 2000). In the first step, the input sentence is partially analyzed at a chunk level based on CFG rules, and then to a clause level based on an ID rule or a leaner precedence rule.
Nojoumian (2003) developed a parser for Persian within the LingBench IDE^{TM} system which functions in two levels: morphology and syntax. The LingBench IDE^{TM} system is a language independent parser which works based on the combination of Recursive Transition Network (RTN), Finite State Automata (FSA), and ATN. In this study, morphological and syntactic rules are defined for the system.
Montazeri et al. (2006) developed a robust rule-based top-down parser which uses the Viterbi algorithm (Viterbi, 1967) on bi-grams recognized as 'bi-rules'. The developed parser is further expanded and used for parsing n-best hypotheses of a speech recognition system (Momtazi et al., 2007).
Dehdari and Lonsdale (2008) developed a parser based on LinkG. In another study of implementing LinkG for Persian, Sajjadi and AbdollahzadeBarforoush (2009) developed a parser that covers a fragment of Persian grammar.
Ayat (2002) developed a rule-based parser for Persian based on HPSG. Moreover, Niknejad (2008) implemented a rule-based HPSG parser enriched with MRS (Copestake et al., 2006) for a Persian–English machine translation.
Ghayoomi and Guillaume (2009) implemented a rule-based parser to analyze nominal and adjectival phrases based on the Interaction Grammar (Perrier, 2000). They reported that LEOPAR, the parser of the Interaction Grammar, can parse all samples of the provided test suite according to the defined rules.
Müller (2010) and Müller and Ghayoomi (2010) implemented a fragment of Persian grammar based on HPSG in the TRALE system (Penn, 2004). The grammar contains a core grammar shared among several languages (Müller,

2013). Bahrani et al. (2011) developed a rule-based parser based on GPSG.

Arabsorkhi et al. (2009) used a semi-supervised approach to induce Persian grammar by employing the Genetic algorithm (Mitchell, 1998). Faili (2006) proposed a history-based inside-outside algorithm which is an extension on the original Inside-Outside algorithm (Lari and Young, 1990) for unsupervised grammar induction. The proposed model has been applied on Persian and English. Mirroshandel and GhassemSani (2008) extended the Constituent Context Model (Klein and Manning, 2002), an unsupervised parsing model, and used the context history and the constituent information of parents for inducing the Persian grammar automatically.

3.5 Summary

In this chapter, we mainly reviewed the theoretical background and the previous studies on well-known grammar formalisms, especially HPSG and the Persian HPSG. Furthermore, we discussed treebanking, parsing, and the parsing evaluation metrics.

Chapter 4

Annotation Scheme for HPSG-based Treebanking

4.1 Introduction

In this chapter, we mainly focus on the annotation scheme required for developing our treebank. Contrary to the normal HPSG that was described in Section 3.2.3, no feature structures are used in our treebank, but the basic properties of the formalism are simulated. Consequently, we call our developed treebank an HPSG-based treebank rather than an HPSG treebank. The annotation scheme that is employed in the development of our treebank is defined based on the basic properties of HPSG.

This chapter contains eight sections. The annotation task and the order of realizing the dependency relations between the constituents at the phrasal and clausal levels are explained in Section 4.2. The elements of the lexical and phrasal sets used in the treebank are defined in Section 4.3. The employed annotation scheme for phrasal and clausal elements are discussed in Sections 4.4 and 4.5. To make the descriptions precise, the tree analyses of the samples for the target syntactic constructions are displayed. Sections 4.6 and 4.7 address two linguistic phenomena that make the annotation of sentences very difficult, namely ellipsis and discontinuity. The chapter is summarized in Section 4.8.

4.2 Annotation Scheme

In the annotation process of a treebank, the annotation scheme plays a major role in producing a standardized result. This scheme can be developed either based on or independent of a linguistic formalism. For our treebank, we employ the annotation scheme designed for the BulTreeBank (Simov, 2003; Osenova and Simov, 2003, 2004), which relies on HPSG (Pollard and Sag, 1994). We adapt the BulTreeBank annotation scheme for Persian. In the designed annotation scheme, no feature structures are used, and the basic properties of HPSG are simulated.

The annotation scheme employed in the BulTreeBank involves the characteristics of both constituency hierarchies and dependency relations for representing the HPSG sort hierarchy and principles via feature graphs to represent the linguistic objects (Simov, 2003). A feature graph is a finite signature defined in King's SRL (King, 1989) to represent an HPSG grammar. According to King's SRL, a natural language is recognized as a set of objects that should be interpreted, and the grammar of the language is essentially a set of description of this set of objects. In the annotation scheme of the BulTreeBank, the feature graphs defined as mother-daughter relations in the constituent structure of signs are the basic representations of a sentence analysis. The constituent structures enriched with the mother-daughter relations are similar to normal phrase structure trees.

The lexical or phrasal labels on the nodes correspond to the categories of the signs, either *word* or *phrase*. The head-daughter relations (dependency relations) are represented as functional labels beside the constituents' labels. The labels that represent the dependency relations are: '*A*' for the Head-Adjunct relation, '*C*' for the Head-Complement relation, '*S*' for the Head-Subject relation, and '*F*' for the Head-Filler relation. Each word has an argument structure in which the number and the type of the required elements are defined. The words which function as a head play a very important role in constructing a syntactic structure and in assigning an appropriate label to the constituent. The analyses provided for the sentences are based on the words' linear appearance, but the analyses might clash because of the extraposition of the constituents' elements. The extraposed constituents are labeled '*DiscE*' (Discontinuous Extraposition). In other words, this label is used for the displaced elements which belong to the argument structure of the head, and they are dominated by the head. The Head-Filler relation is defined at the topmost node to bind the extraposed elements. Contrary to Keller (1995), in our annotation scheme, both extraposing and fronting are analyzed with one operation, and they are bound with the Head-Filler-Schema. The empty node '*nid*' (non immediate dominance) defines the canonical positions of the extraposed elements, and co-referential indexes are used for linking *nid*s to the extraposed elements. In fact, the *nid* node functions as a trace in the analyses. The derived trees according to this scheme are projective.

Based on the sort hierarchy defined in Osenova and Simov (2004), the sort

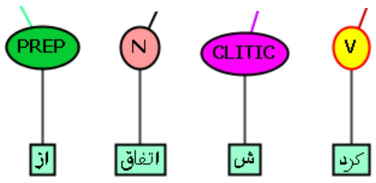


Figure 4.1: Graphical representation of lexical elements

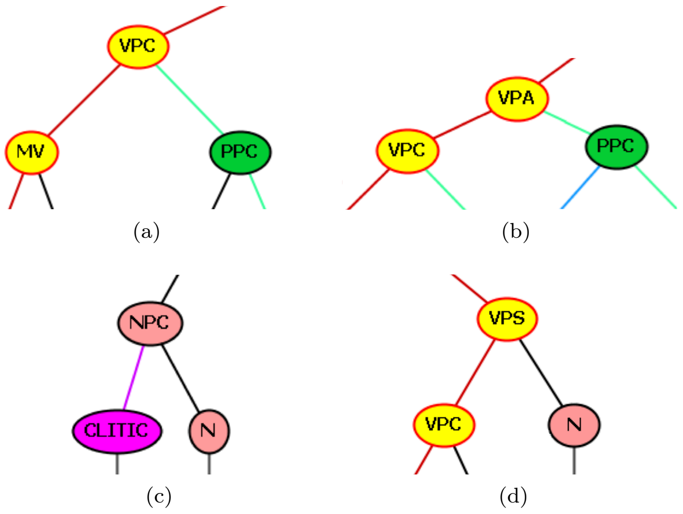


Figure 4.2: Graphical representation of phrasal elements

sign can be either a *word* or a *phrase*. The *word* sign is unary branching with a lexical item as its sole daughter, as displayed in Figure 4.1. The daughters of the *phrase* sign can be words or phrases, as demonstrated in Figure 4.2. This fragments of trees should be read right-to-left, because Persian is a right-to-left language. The *phrase* sign can contain phrasal elements in Figures 4.2a and 4.2b, lexical elements in Figure 4.2c, or the combination of the two in Figure 4.2d. In addition to these signs, the unary branching functional elements, namely *nid*, *DiscE*, *Pragmatic*[1], *S*, and *ROOT* elements are added to the *sign* set. Additionally, a set of unary branching elements that determines the type of clausal saturated verb phrases is added to the *sign* set in the BulTreeBank.

The head-dependency realizations are governed by the ID Schemas, and they are restricted to the below sequence throughout the development of the BulTreeBank (Simov and Osenova, 2003, Sec. 3), i. e. the schemas are prioritized according to this sequence to combine the elements and to determine the type of the head-daughter dependency relations:

$$Complement \rightarrow Subject \rightarrow Adjunct$$

[1]It is a functional label to indicate that this element has a pragmatic function.

This sequence recognized as a strong constraint means that the realization of obligatory arguments (complements) other than the subject has to precede the realization of the Head-Subject relation, and the realization of the obligatory arguments in general has to precede the realization of Head-Adjunct relation. The Head-Filler relation is only used for binding the extraposed elements which belong to the argument structure of a lexical element. It should be added that this constraint for prioritizing the recognition of the dependencies does not exist in HPSG where a collection of constraints without a specific order of application is defined.

As mentioned in Section 3.3, there are two goals behind the development of a treebank. If the end users of the data source are computational linguists who aim at processing the language automatically for applications like parsing, the annotation of the data should be simple with minimum complexity, and it should be limited to the linear word order. If the end users are linguists who study linguistic theories and test hypotheses, the data source should contain information which allows them to search for a specific linguistic phenomenon in the corpus. Consequently, a rich data source should be provided in such a way to meet the needs of the two groups of investigators.

The constraint proposed by Simov and Osenova (2003, Sec. 3) to prioritize combining the elements is harmless for languages with fixed constituent order. As a result, the developed data set based on this constraint can be used for both computational and linguistic applications. Using this constraint for languages with free or relatively free constituent order, however, increases the degree of the annotation complexity. The reason for increasing the degree of complexity is that the number of structures with extraposition and scrambling increases. Since the canonical positions of the extraposed and scrambled elements are determined by traces in the syntactic analyses to follow this constraint, the number of traces increases. Consequently, the applicability of the data source for computational applications will be limited. Moreover, the decision on the canonical position of constituents, such as prepositional phrases is ambiguous and undecidable. The advantage of providing such analyses is useful for linguistic investigations which allow linguists to search for a specific phenomenon in the target language and to find out how frequent the phenomenon is in the target language. Since we aim at developing a multipurpose data set to be used for computational and linguistic purposes, we modify the proposed constraint and adapt the constraint according to the requirements of the Persian language.

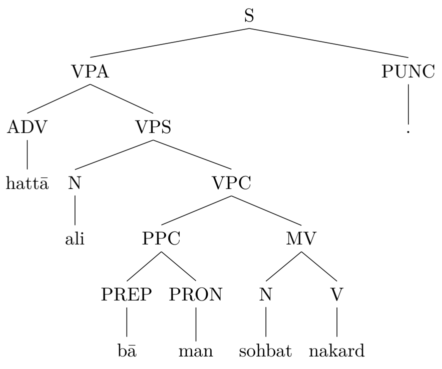
Since in Persian, a subject, complement, or topic might drop, we add three more specific relations to the list of the dependency relations to cover these phenomena. The complete list of the recognized dependency relations in our treebank for Persian are the Head-Subject, Head-Complement, Head-Adjunct, Head-Filler, Head-Subject-Drop (*SD*), Head-Complement-Drop (*CompD*), and Head-Topic-Drop (*TD*) relations.

There is a class of adjunct elements, such as adverbs and conjunctions, which have wide scope and cover the whole sentence rather than the local context. For instance, the scope of the adverb 'حتی' /hatta/ 'even' in Example 4.1 is the whole sentence as represented in the tree diagram. These elements are realized after the realization of the Head-Subject relation.

(4.1) حتی علی با من صحبت نکرد.

hattā	ali	bā	man	sohbat	na-kard
even	Ali	with	I	talk	NEG-did.3SG

'Even Ali did not talk to me.'



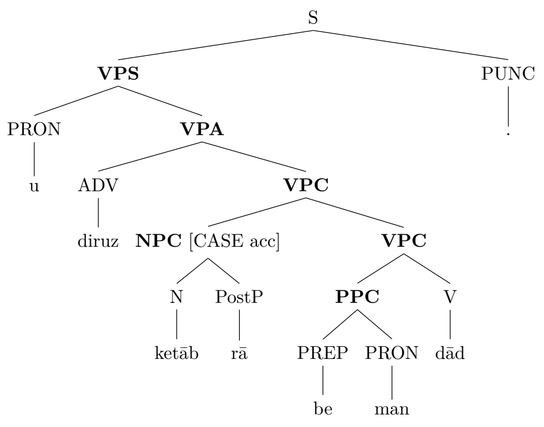
We modify the constraint proposed by Simov and Osenova (2003, Sec. 3) and adapt it for the Persian treebank to prioritize realization of the dependency relations:

$$\left\{\frac{\dfrac{Complement}{Complement - Drop}}{Adjunct}\right\} \rightarrow \left\{\frac{\dfrac{Subject}{Subject - Drop}}{Topic - Drop}\right\} \rightarrow Filler \rightarrow Adjunct$$

This constraint means that the realization of either a Head-Complement or Head-Adjunct relation has to precede the realization of a subject, a dropped subject, or a dropped topic. If the adjunct has wide scope, it is realized after the realization of the Head-Subject relation. For discontinuous constructions, the Head-Filler relation is applied solely after the realization of the Head-Subject relation. Since the constituent order in Persian is relatively free and the elements might scramble, we make the realization of the Head-Complement and Head-Adjunct relations flexible, and either of the relations can be realized based on their linear appearance in a sentence. It should be mentioned that the defined head-daughter relations in our treebank rely mainly on the HPSG formalism introduced by Pollard and Sag (1994) and the BulTreeBank experience (Osenova and Simov, 2003). This set of head-daughter relations differs slightly from the head-daughter relations implemented in the Persian HPSG by Müller (2010) and Müller and Ghayoomi (2010). In their implementation the Head-Subject-Schema does not exist among the head-daughter relations, but in our annotation scheme we have it. Moreover, we do not use the Head-Cluster-Schema defined in their grammar.

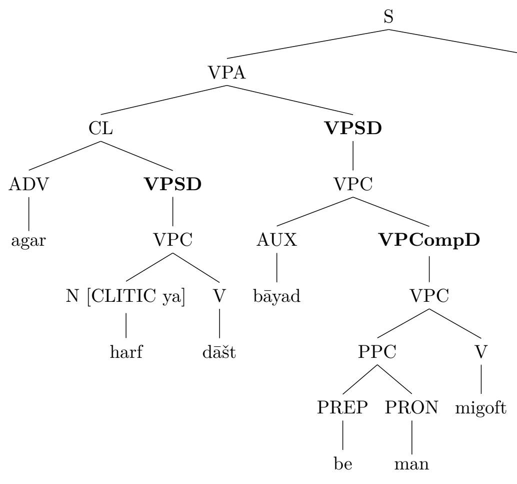
The tree diagram in Example 4.2 shows how the head-dependency relations are realized. As can be seen, first the Head-Complement relation is realized, then the Head-Adjunct relation, and finally the Head-Subject relation.

(4.2) او دیروز کتاب را به من داد.
u diruz ketāb rā be man dād
he/she yesterday book DOM to me gave.3SG
'He/she yesterday gave the book to me.'



The tree diagram in Example 4.3 represents how the head-dependency relations for Subject-Drop and Complement-Drop are realized.

(4.3) اگر حرفی داشت باید به من می‌گفت.
agar harf-i dāšt bāyad be man mi-goft
if word-INDEF had.3SG should to me IMPF-said.3SG
'If he/she had a word, he/she should have said [it] to me.'

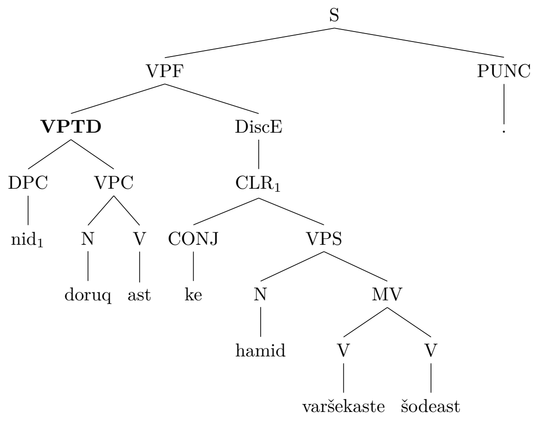


The tree diagram in Example 4.4 shows the head-dependency relation for Topic-Drop.

(4.4) دروغ است که حمید ورشکسته شده‌است.

doruq	ast	ke	hamid	varšekaste	šode-ast
lie	is	that	Hamid	bankrupt	become-3SG

'[It] is a lie that Hamid is bankrupted.'



Tabibzadeh (2012, pp. 10–13) has a finer classification of head-dependency relations such that complements can be obligatory or optional, and adjuncts can be general or specific. In this view, complements belong to the argument structure of the head, and the head controls the number and the type of the arguments, while adjuncts do not belong to the argument structure of the head, and the head has no control over them. To make the obligatory complements distinct, Tabibzadeh expresses that the lack of these elements makes the sentence ungrammatical, but optional complements do not have such a property. Specific adjuncts which have narrow scope have a special semantic relation with the head, and any kind and any number of them can be used in any order. General adjuncts, on the other hand, have wide scope and they modify a constituent that dominates the maximal projection in which it occurs. Example 4.5 from Tabibzadeh (2012, p. 12) illustrates these four types of head-dependencies. In this example, the head of the sentence is the verb 'تحویل داد' /tahvil dād/ 'delivered'. The obligatory complements are 'علی' /ali/ 'Ali' and 'کتاب را' /ketāb rā/ 'the book', and the optional complement is 'به من' /be man/ 'to me'. The general adjunct is 'دیروز' /diruz/ 'yesterday', and the specific adjunct is 'با میل' /bā meyl/ 'with pleasure'.

(4.5) علی دیروز کتاب را با میل به من تحویل داد.

ali	diruz	ketāb	rā	bā	meyl	be	man	tahvil	dād
Ali	yesterday	book	DOM	with	pleasure	to	I	delivery	gave.3SG

'Ali yesterday delivered the book to me with pleasure.'

Tabibzadeh (2012, pp. 339–343) believes that there is an order, but not necessarily a strict order, between the dependents and the heads, and changing the order relatively does not effect the dependency relations. By contrast, there is a strict order between the dependents and the heads in noun phrases (Tabibzadeh, 2012, p. 39) such that the post-nominal elements are realized before the pre-nominal elements (Tabibzadeh, 2012, p. 23). This is the order of realizing the dependency relations in Tabibzadeh (2012):

$$\begin{gathered} \textit{general adjunct} \rightarrow \textit{specific adjunct} \rightarrow \\ \textit{optional complement} \rightarrow \textit{obligatory complement} \rightarrow \textit{head} \end{gathered}$$

Contrary to Tabibzadeh (2012), in the current study, we have a more coarse-grained classification of dependency relations. If a lexical item contains an obligatory element in its argument structure, this element is recognized as a complement or subject, and the other elements (optional complements, general or specific adjuncts) are recognized as adjuncts. The lack of an obligatory element makes the sentence ungrammatical. To make the data usable for computational applications, we treat general and specific adjuncts equally without using specific labels for them in the process of data annotation.

In the following section, the properties of the annotation scheme used for Persian treebanking are described.

4.3 Lexical and Phrasal Elements

In the original data set of the Bijankhan Corpus (Bijankhan, 2004) used for our study, 14 main POS tags are defined, and clitics are not recognized as individual elements. Since possessive and object clitics as well as copulative verbal clitics play syntactic roles in a sentence, we split them from their hosts and assign each split-off element a POS tag to represent a more accurate syntactic analysis of the sentence. We assign these clitics the ‘clitic’ label, which results in an increase in the number of the main syntactic categories to 15 POS tags. Other types of clitics in which their morpho-syntactic properties are defined in the POS tag of the host remain unchanged as they are in the Bijankhan Corpus.

The labels of the constituents are realized based on the syntactic category of the head in the local context, such as adjective (ADJ), adverb (ADV), classifier (CLASS), clitic (CLITIC), conjunction (CONJ), interjection (I), noun (N), number (NUM), preposition (PREP), pronoun (PRON), post-position (PostP), punctuation (PUNC), residual (RES). There are three subtypes of determiners in the Bijankhan Corpus: a demonstrative (DEM), an interrogative word as a determiner (Q), or a quantifier as a determiner (DET). There are four subtypes of verbs in the Bijankhan Corpus: the auxiliary (AUX), infinitive form (INFV),

Table 4.1: Dependency relations for phrasal elements

Phrasal Element	Types of Relations						
	Adjunct	Complement	Filler	Subject	Subject-drop	Topic-drop	Complement-drop
ADJP	ADJPA	ADJPC					
ADVP	ADVPA	ADVPC					
CP		CPC					
DP	DPA	DPC					
IP	IPA	IPC					
NP	NPA	NPC	NPF				
PP	PPA	PPC					
VP	VPA	VPC	VPF	VPS	VPSD	VPTD	VPCompD

past-participle form (PPARV), or conjugated form (V). Each of these lexical labels contains morpho-syntactic and semantic information that is described more in detail in the pre-processing step in Section 5.4.1.

In addition to these lexical labels, the complex lexical elements are recognized as well based on their heads in the local context, such as a compound adjective (MADJ), compound adverb (MADV), compound conjunction (MCONJ), compound determiner (MD), compound noun (MN), compound preposition (MP), and compound verb (MV). In the structure of phrasal elements, we treat complex lexical elements in a similar way as simple lexical elements.

There are eight phrasal elements that are realized with respect to the syntactic role of the head, such as an adjectival phrase (ADJP), an adverbial phrase (ADVP), an interjection phrase (IP), a coordination phrase (CP), a determiner phrase (DP), a noun phrase (NP), a prepositional phrase (PP), and a verb phrase (VP). Table 4.1 summarizes the set of phrasal elements enriched with the dependency relations.

Similar to the BulTreeBank, we use unary branching to represent clauses as saturated verb phrases and label the nodes with respect to the type of the clause. In the Persian treebank, we label clauses as follows: relative clause (CLR), reduced relative clause that is a relative clause without the relativizer (CLRR), free relative clause (CLFR), complement clause (CLC), complement clause with a dropped complementizer (CLCD), and an interrogative clause (CLQ). The label CL is used for all other types of clauses.

In the next section, we describe the syntactic construction of the phrasal elements and clauses, and provide examples for them along with their syntactic analyses represented as tree diagrams.

4.4 Syntactic Construction of Phrasal Elements

In this section, we exemplify various constituents described in the previous section and represent the dependency relations in tree diagrams. In Persian syntax, most of the constituents are head first. Verb phrases are head-last because of the

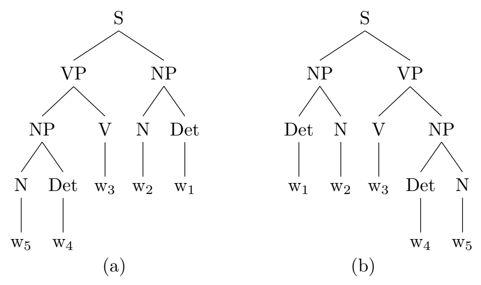


Figure 4.3: Right-to-left (a) and left-to-right (b) representation of the sentence w_1 w_2 w_3 w_4 w_5

SOV constituent order, and noun phrases are conditioned to use pre-nominal or post-nominal elements in a strict order. If the head adjective of an adjectival phrase is modified with an adverb, the phrase is head last. If the head adjective takes elements as its argument, the phrase is head first. Since Persian is a right-to-left language, the trees should also be represented right-to-left. To be readable, we provide the transliterations and glosses of the Persian words in the examples. Consequently, to be consistent, we represent the trees left-to-right with transliterated Persian words along with the constituent labels. Changing the direction of the tree does not have any effect on the order of dominance relations. Therefore, in our study, the left-to-right tree representation in Figure 4.3b is used instead of the right-to-left tree representation in Figure 4.3a.

As displayed in Tables B.1 and B.2 of Appendix B, in the Bijankhan Corpus the post-nominal indefinite marker 'ی' /-i/ and Ezāfe are encoded in the POS tags of the words as clitics. In the displayed tree diagrams, we do not provide the detailed POS tags of the words, but rather their main POS tags. To represent clitics in the trees, the feature 'CLITIC' is used with the value {empty|ezāfe|ya} where the value 'empty' expresses no usage of clitic, the values 'ezāfe' and 'ya' express the usage of Ezāfe and the indefinite marker 'ی' /-i/ respectively as represented in the tree diagram of Example 4.3 on page 74. Since the value 'empty' of the feature 'CLITIC' is very redundant, we do not add this feature to all nodes on the tree diagrams.

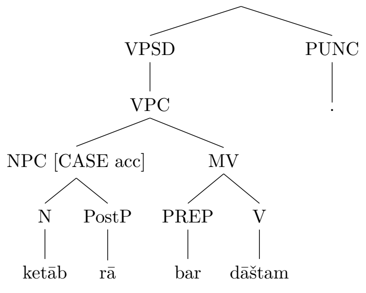
4.4.1 Verb Phrase

Argument Structure

Verbs play the most important role in a sentence. The lexical property of each verb determines the number and the type of arguments in its subcategorization frame, and violating the verb's lexical requirement makes a sentence ill-formed. In Persian, there exist four types of verbal complements: (a) noun phrases or determiner phrases marked with the particle 'را' /rā/ as direct objects, (b) prepositional phrases as indirect objects, (c) clausal complements marked with the complementizer 'که' /ke/ 'that', and (d) noun phrases as bare complements without the direct object marker, or noun phrases as bare complements with the post-nominal indefinite determiner 'ی' /-i/. Since Persian has SOV constituent order, the default positions of direct and indirect objects and bare complements in a sentence are before the head verb, but clausal complements might follow the head verb. It is possible to change the positions of direct and indirect objects and bare complements only. Changing the position of clausal complements makes the sentence ill-formed. In predicative sentences in which copulas are the heads, adjectival phrases, adverbial phrases, noun phrases, numbers, or prepositional phrases are recognized as the predicate complements. To determine the type of the dependency relations, the linear order of constituents is taken into consideration.

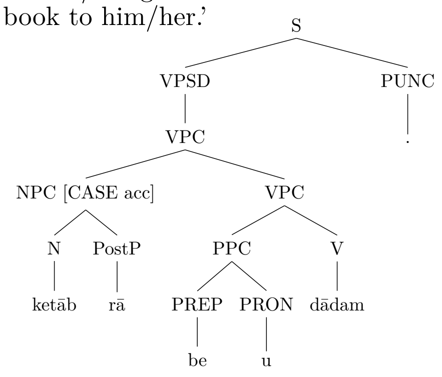
In Example 4.6, the noun 'کتاب' /ketāb/ 'book' is marked with the direct object marker 'را' /rā/. Since this marker is essential to realize the constituent as the direct object, we determine the type of the dependency relation as complement among our four types of dependency relations. The verb 'بر داشتن' /bar dāštan/ 'pick up' is a compound verb composed of the particle 'بر' /bar/ 'on' and the light verb 'داشتن' /dāštan/ 'have'. Particles are considered as prepositions in the Bijankhan Corpus. We keep its assigned POS tag in our analyses. This verb is realized as a compound verb under the node 'MV'. The verb 'بر داشتن' /bar dāštan/ 'pick up' requires a complement and a subject in its argument structure. In this sentence, the noun 'کتاب' /ketāb/ 'book' marked with the direct object marker 'را' /rā/ is realized as the direct object. Since the subject is dropped, the Head-Subject-Drop-Schema is employed such that the unary branching node labeled 'VPSD' is used for filling the subject argument, and the node shares lexical properties retrieved from the verb conjugation.

(4.6) کتاب را بر داشتم.
ketāb rā bar ds̄št-am
book DOM PART had-1SG
'I picked up the book.'



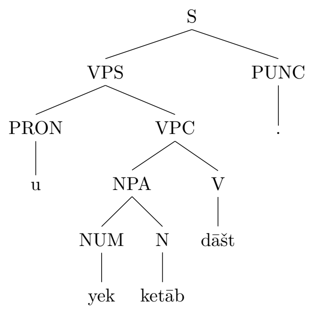
In Example 4.7, the verb ‘دادن’ /dādan/ ‘give’ requires two objects (a direct object and an indirect object) and a subject in its argument structure. The noun ‘کتاب’ /ketāb/ ‘book’ is marked with the direct object marker and it is realized as the direct object, and the prepositional phrase is realized as the indirect object. The unary branching node labeled ‘VPSD’ is used for filling the subject argument, and the node shares lexical properties retrieved from the verb conjugation.

(4.7) کتاب را به او دادم.
ketāb rā be u dād-am
book DOM to he/she gave.1SG
‘I gave the book to him/her.’



In Examples 4.8 and 4.9, the verb ‘داشتن’ /dāštan/ ‘have’ requires either a noun or a determiner phrase as its complement and a subject in its argument structure. The complement does not have the direct object marker, but it appears with an indefinite determiner. The lexeme ‘یک’ /yek/ ‘a, an, one’ can function both as a number and an indefinite determiner, but in the Bijankhan Corpus no distinction is made between them and they are assigned the POS tag ‘NUM’ (number), and the post-nominal indefinite marker ‘ی’ /-i/ is encoded as a clitic in the POS tag of the host. In the annotation of the sentences, we keep the original POS tag.

(4.8) او یک کتاب داشت.
u yek ketāb dāšt
he/she one book had.3SG
‘He/she had a book.’

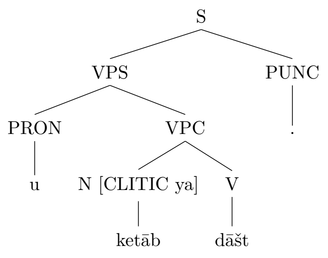


(4.9) او کتابی داشت.

u ketābi dāšt

he/she book.INDEF had.3SG

‘He/she had a book.’



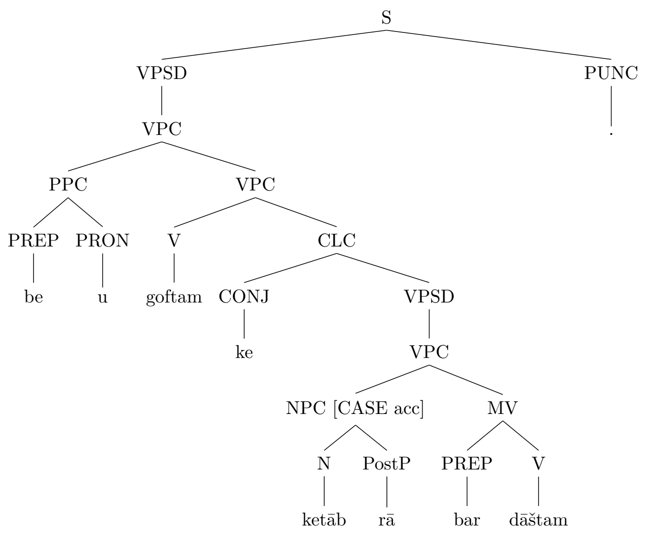
In Example 4.10, the verb ‘گفتن’ /goftan/ ‘say, tell’ requires a subject, an indirect object, and a clausal complement in its argument structure. The prepositional phrase headed with the preposition ‘به’ /be/ ‘to’ is realized as its complement, and the clause marked with the complementizer ‘که’ /ke/ ‘that’ is realized as its second complement. We use the label ‘CLC’ to determine that the subordinate clause is marked with the complementizer and the clause is realized as the complement of the verb. The unary branching node labeled ‘VPSD’ is used for filling the subject argument, and it shares lexical properties retrieved from the verb conjugation.

(4.10) به او گفتم که کتاب را بر داشتم.

be u goft-am ke ketāb rā bar dāšt-am

to he/she said-1SG that book DOM PART had-1SG

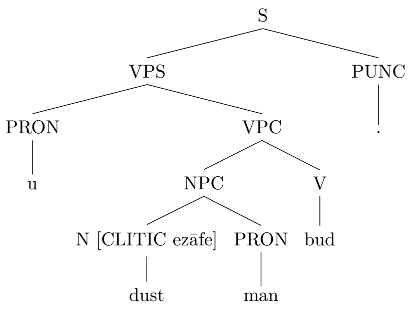
‘I said to him/her that I picked up the book.’



In Example 4.11, the verb ‘بودن’ /budan/ ‘be’ is a copula and it always requires a predicative complement and a subject in its argument structure.

The potential predicative complements are adjectival phrases, adverbial phrases, prepositional phrases, and noun phrases. In Persian, noun phrases, or determiner phrases that are not marked with 'را' /rā/ and agree with the verb in person and number are recognized as the sentential subject. When the overt subject is used in the sentence, the Head-Subject relation is employed and the label 'VPS' is assigned to the mother node.

(4.11) او دوست من بود.
u dust.e man bud
he/she friend.EZ I was
'He/she was my friend.'



The Light Verb Construction

There are simple and compound verbs in Persian. There are a number of studies on the construction of compound verbs in Persian, such as Karimi (1997), DabirMoghaddam (1997), KarimiDoostan (1997), Tabatabayi (2005), Gerdes and Samvelian (2008), Müller (2010), Bagherbeygi and Shamsfard (2012), Salehi et al. (2012), and Taslimipoor et al. (2012).

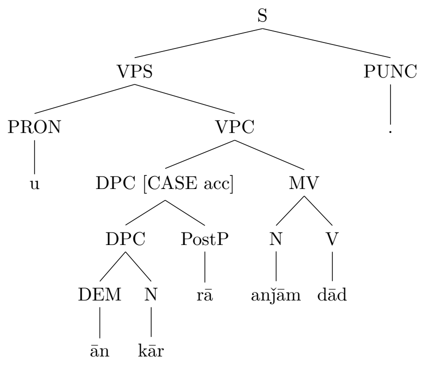
Compound verbs are composed of a pre-verbal element and a light verb. The distinction between compositional or non-compositional light verb constructions is out of the scope of our study, but in the annotation it should be determined minimally that the pre-verbal and light verb elements form a compound verb. Therefore, we do not use the Head-Cluster-Schema in (Müller, 2010; Müller and Ghayoomi, 2010) for light verb constructions. The most frequent light verbs are (Kalbassi, 2001):

'آمدن' /āmadan/ 'come'	'خواندن' /xāndan/ 'read'	'شدن' /šodan/ 'become'
'آوردن' /āvardan/ 'bring'	'خوردن' /xordan/ 'eat'	'کشیدن' /kešidan/ 'pull'
'افتادن' /oftādan/ 'fall'	'دادن' /dādan/ 'give'	'کردن' /kardan/ 'do'
'انداختن' /andāxtan/ 'drop'	'داشتن' /dāštan/ 'have'	'گذاشتن' /gozāštan/ 'put'
'بردن' /bordan/ 'take'	'دیدن' /didan/ 'see'	'گرفتن' /gereftan/ 'get'
'بستن' /bastan/ 'close'	'رفتن' /raftan/ 'go'	'گفتن' /goftan/ 'say, tell'
'پاشیدن' /pāšidan/ 'disperse'	'زدن' /zadan/ 'hit'	'یافتن' /yāftan/ 'find'

In the annotation of compound verbs, we use the label ‘MV’ to express that the two elements are treated and annotated as a compound verb. Whether there is a long distance between pre-verbal elements and the light verbs, there are two syntactic constructions for compound verbs. If the compound verbs are more lexicalized, they are less splittable as is the case with the noun ‘انجام’ /anǰām/ ‘perform’ and the light verb ‘دادن’ /dādan/ ‘give’ in Example 4.12.

(4.12) او آن کار را انجام داد.

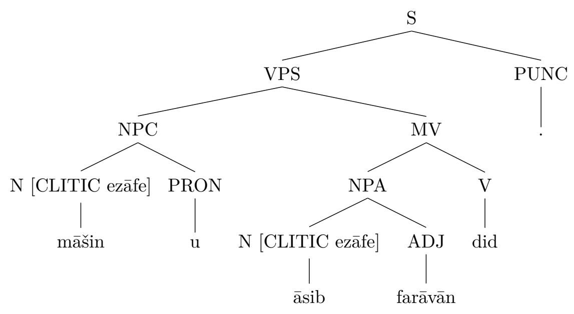
u ān kār rā anǰām dād
he/she that task DOM perform gave.3SG
‘He/she did that task.’



If the compound verbs are less lexicalized, there might be a long distance between the pre-verbal element and the light verb. In these cases, the pre-verbal element dominates the intervening elements as represented in the tree analyses of Examples 4.13 and 4.14 where the nouns ‘آسیب’ /āsib/ ‘damage’ and ‘آماده’ /āmāde/ ‘ready’ are split from the light verbs ‘دیدن’ /didan/ ‘see’ and ‘کردن’ /kardan/ ‘do’, respectively. In the tree diagram of Example 4.13, the noun ‘آسیب’ /āsib/ ‘damage’ is modified by the adjective ‘فراوان’ /farāvān/ ‘a lot’, which is appeared between the pre-verbal element and the light verb. Since the pre-verbal element is marked with Ezāfe, we annotated it as the post-nominal modifier of the noun. As a result, the label ‘NPA’ is assigned for combining the head noun and its modifier. In the tree diagram of Example 4.14, the pre-verbal element requires a complement and satisfying the lexical requirement has caused the distance between the pre-verbal element and the light verb.

(4.13) ماشین او آسیب فراوان دید.

māšin-e u āsib-e farāvān did
car-EZ he/she damage-EZ a.lot saw.3SG
‘His/her car was damaged a lot.’

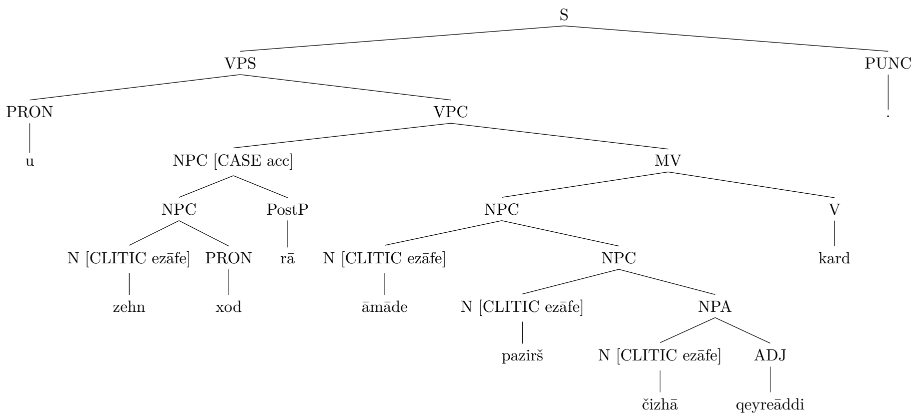


(4.14) او ذهن خود را آماده پذیرش چیزهای غیرعادی کرد.

u	zehn-e	xod	rā	āmāde-ye	pazirš-e	čiz-hā-ye
he/she	mind-EZ	self	DOM	ready-EZ	acceptance-EZ	thing-PL-EZ

qeyre-āddi	kard
un-usual	did.3SG

‘He/she made his/her mind ready for accepting unusual things.’



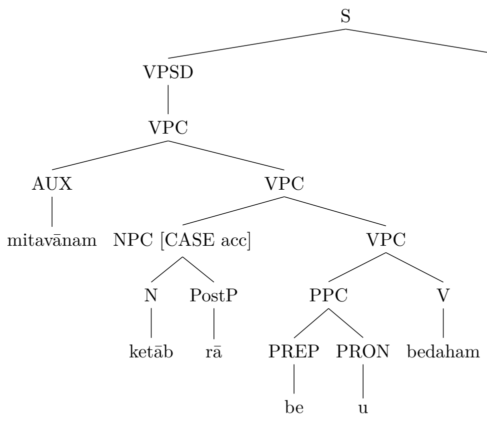
Modal Verbs

There are three groups of modal verbs in Persian described in Section 2.3. The default position of modals is after the subject and before the main verb. A Head-Complement relation exists between the head modal verb and the sister constituent, as in Example 4.15.

(4.15) می‌توانم کتاب را به او بدهم.

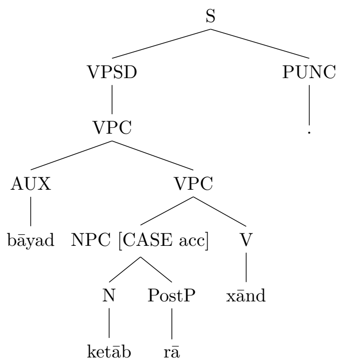
mi-tavān=am	ketāb	rā	be	u	be-dah-am
IMPF-can-1SG	book	DOM	to	he/she	SUBJ-give-1SG

‘I can give the book to him/her.’



One property of Persian modal verbs is that they can be impersonal; i. e. in these syntactic constructions, there is no overt subject, but the verb is conjugated third person singular, as in Example 4.16. Since the verb is conjugated, a subject has to be realized to saturate the elements in the argument structure of the verb and to complete the syntactic analysis of the sentence. In such syntactic constructions, we assume that the subject is dropped and the label 'VPSD' is used to fill the subject position.

(4.16) باید کتاب را خواند.
bāyad ketāb rā xānd
must book DOM read.3SG
'One must read the book.'



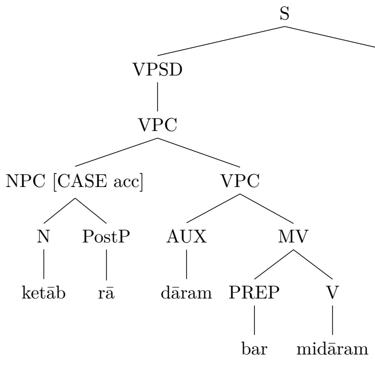
Auxiliary Verbs

Auxiliaries like 'داشتن' /dāštan/ 'have' and 'بودن' /budan/ 'be' in periphrastic constructions are used for progressive and perfect aspects respectively, and the prefix 'می‌ـ' /mi-/ is used for the imperfect aspect. The auxiliary 'داشتن' /dāštan/ 'have, has' comes before the main verb and it conjugates for person and number. Following the study of Bonami and Samvelian (2009), we use the Head-Complement relation for combining the head auxiliary and the sister constituent as presented in Example 4.17.

(4.17) کتاب را دارم بر می‌دارم.

ketāb	rā	dār-am	bar	mi-dār-am
book	DOM	have-1SG	PART	IMPF-have-1SG

'I am picking up the book.'

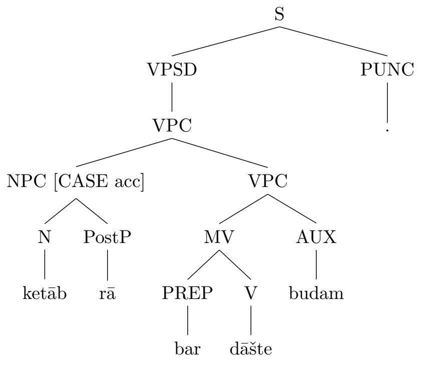


The auxiliary 'بودن' /budan/ 'be' as the syntactic head comes after the main verb and it conjugates for person and number. If the main verb in the perfect aspect is compound, first the main verb and the pre-verbal element combine to form a compound verb, and then they combine with the auxiliary verb, as in Example 4.18.

(4.18) کتاب را بر داشته بودم.

ketāb	rā	bar	dāšte	bud-am
book	DOM	PART	had	was-1SG

'I had picked up the book.'



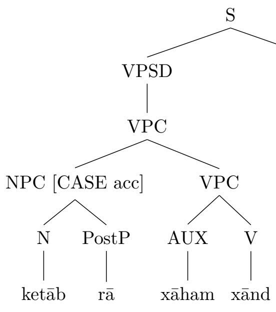
Tense, Mood, and Negation

The present and past tenses are expressed lexically because in Persian verbs have present and past stems. To express the future tense, the auxiliary verb 'خواستن' /xāstan/ 'will, would' is used. This auxiliary verb comes before the main verb and conjugates for person and number. The Head-Complement relation is used for such syntactic constructions as presented in the tree diagram of Example 4.19.

(4.19) کتاب را خواهم خواند.

ketāb	rā	xāh-am	xānd
book	DOM	will-1SG	read.3SG

'I will read the book.'

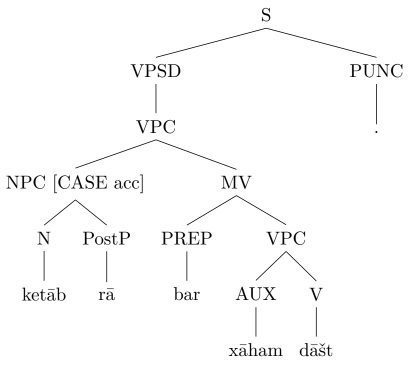


If the main verb is a compound verb, the auxiliary verb can appear between the pre-verbal element and the main verb, such as Example 4.20.

(4.20) کتاب را بر خواهم داشت.

ketāb	rā	bar	xāh-am	dāšt
book	DOM	PART	will-1SG	had.3SG

'I will pick up the book.'

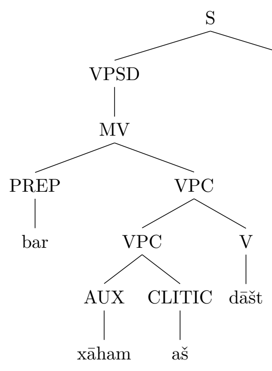


If the object clitic is between the auxiliary and the main verb, first the Head-Complement relation is used for combining the auxiliary and the clitic (Müller, 2010, Sec. 5). Then, the phrase headed by the auxiliary combines with the main verb. Finally, the phrase combines with the particle to form a compound verb, as in Example 4.21. In our annotation, we split clitics from their hosts described in the pre-processing step in Section 5.4.1.

(4.21) بر خواهمش داشت.

bar	xāh-am=aš	dāšt
PART	will-1SG=3SG	had.3SG

'I will pick it up.'

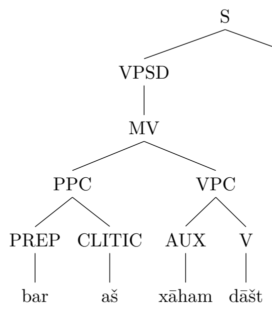


If the object clitic is after the particle, first the Head-Complement-Schema is used for combining the auxiliary and the verb. Then, this relation is re-used for combining the particle and the clitic, and the label 'PPC' is assigned to the node, as in Example 4.22. The two nodes combine together, and the node is labeled 'MV'.

(4.22) برش خواهم داشت.

bar=aš	xāh-am	dāšt
PART=3SG	will-1SG	had.3SG

'I will pick it up.'

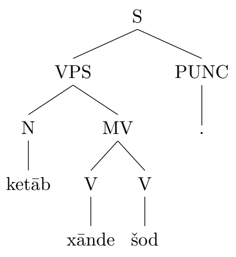


The subjunctive and imperative moods are expressed by the prefix 'بـ' /be-/ attached to the verb. It should be added that it is possible for a verb to be used in the subjunctive mood without the subjunctive prefix. Negation of the verbs is expressed by attaching the negation prefix 'نـ' /na-, ne-/ to the verb or the progressive prefix 'میـ' /mi-/. Both mood and negation are treated lexically and they are encoded in the POS tags of the verbs.

Passive and Causative Constructions

Although the argument structure of a verb is fixed, it is possible to change the number of the verb's required argument in passive and causative constructions. Following VahediLangaroodi (1999), we treat the passive construction as a compound verb construction, and we use the label 'MV' in the annotation, as represented in the tree diagram of Example 4.23.

(4.23) کتاب خوانده شد.
ketāb xānde šod
book read became.3SG
'The book was read.'

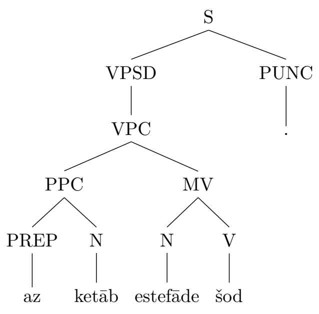


A number of verbs with two arguments like 'استفاده کردن' /estefāde kardan/ 'use' require a prepositional phrase headed with the preposition 'از' /az/ 'of, from, than' as their complements, as in Example 4.24.

(4.24) او از کتاب استفاده کرد.
u az ketāb estefāde kard
he/she from book use did.3SG
'He/she used the book.'

The passivization of such verbs is problematic, because on the one hand the prepositional phrase takes the subject position and it should be recognized as the subject of the sentence, and on the other hand a prepositional phrase cannot be a subject. For these cases, we realize the prepositional phrase as the complement of the passivized verb, and we use the label 'VPSD' as a placeholder for the subject to follow the order of defining the head-daughter schemas. Example 4.24 is passivized in Example 4.25.

(4.25) از کتاب استفاده شد.
az ketāb estefāde šod
from book use became.3SG
'The book was used.'

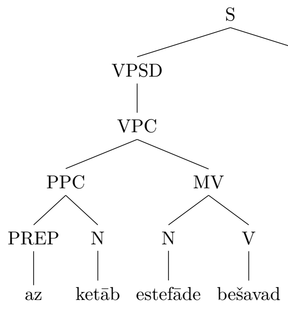


It is possible to have this verb in subjunctive mood as well without an overt subject, as in Example 4.26.

(4.26) از کتاب استفاده بشود.

az	ketāb	estefāde	be-šav-ad
from	book	use	SUBJ-become-3SG

'Use the book./ the book should be used.'



In the causative construction, a complement is added to the verb's argument list. The verb 'خوردن' /xordan/ 'eat' normally requires two arguments, as in Example 4.27. After causitivization of the verb, it requires three arguments as represented in the tree diagram of Example 4.28.

(4.27) او سیب را خورد.

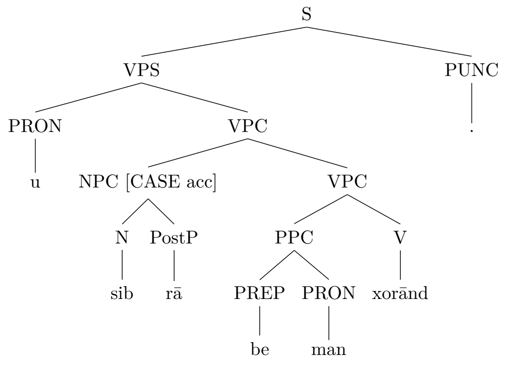
u	sib	rā	xord
he/she	apple	DOM	ate.3SG

'He/she ate the apple.'

(4.28) او سیب را به من خوراند.

u	sib	rā	be	man	xor-ānd
he/she	apple	DOM	to	I	eat-CAUS.3SG

'He/she fed me an apple.'



Raising Construction

Raising verbs are the ones that do not require an external argument, and their subject originates from the embedded clause (Taleghani, 2008, p. 6) such as 'seem' in English. Contrary to Karimi (2005, pp. 12–14) who argues that Persian does not have a raising verb because of the lack of agreement between the subject of the matrix clause and the extraposed embedded clause, such as Example 4.29, Darzi (1996, pp. 93–94) believes that an expletive, such as 'این' /in/ 'this', can be used in raising constructions to fill the subject position, as in Example 4.30.

(4.29) به نظر می‌آید (که) بچه‌ها این فیلم را دیده باشند.
be nazar mi-ā-yad (ke) bače-hā in film rā dide
to view IMPF-come-3SG (that) child-PL this movie DOM seen
bāš-and
be-3PL
'It seems that the children have seen this movie.'

(4.30) این لازم است که علی کتاب را به او بدهد.
in lāzem ast ke ali ketāb rā be u be-dah-ad
this necessary is that Ali book DOM to he/she SUBJ-give-3SG
'It is necessary that Ali give the book to him/her.'

Moyne and Carden (1974) and SoheiliIsfahani (1976) believe that in syntactic constructions like Example 4.30, 'این' /in/ 'this' is the head and it constructs a noun phrase with the clause. Additionally, Moyne and Carden (1974) further express that the sentential subject as a clause is obligatorily extraposed to the post-verbal position. Similar to the idea of Moyne and Carden (1974), Tabibzadeh (2012, p. 79) believes that in such syntactic constructions the elements that are extraposed to the post-verbal position are 'clausal subjects' of the predicative sentences, such as the clause 'که برویم' /ke beravim/ 'that we go' in Example 4.31.

(4.31) لازم است که برویم.
lāzem ast ke be-rav-im
necessary is that SUBJ-go-1PL
'It is necessary that we go.'

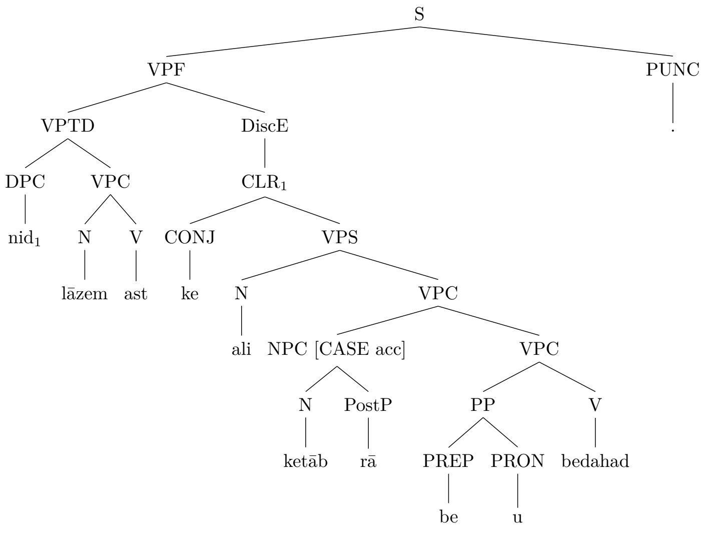
In the annotation of such syntactic constructions, we follow the analysis provided by Moyne and Carden (1974), SoheiliIsfahani (1976), and Tabibzadeh (2012), and we propose that in the topic position the expletive 'این' /in/ 'this' as the head is modified by a relative clause marked with the relativizer 'که' /ke/ 'that'. Consequently, Example 4.30 is rewritten as Example 4.32. If the expletive in the topic position is dropped, the clausal subject will obligatorily extrapose to the post-verbal position as represented in the tree diagram of Example 4.33. We will discuss extraposition more in Section 4.7. Since the main verb of the sentence is conjugated as third person singular, the subject should be retrieved. To represent the topic position that is dropped in these syntactic constructions, the label 'VPTD' is used for filling the subject position and to indicate that the subject of the verb in topic position is an expletive that is dropped.

(4.32) این که علی کتاب را به او بدهد لازم است.
in ke ali ketāb rā be u be-dah-ad lāzem ast
this that Ali book DOM to he/she SUBJ-give-3SG necessary is
'That Ali give the book to him/her is necessary.'

(4.33) لازم است که علی کتاب را به او بدهد.

lāzem	ast	ke	ali	ketāb	rā	be	u	be-dah-ad
necessary	is	that	Ali	book	DOM	to	he/she	SUBJ-give-3SG

'[It] is necessary that Ali give the book to him/her.'



4.4.2 Noun Phrase

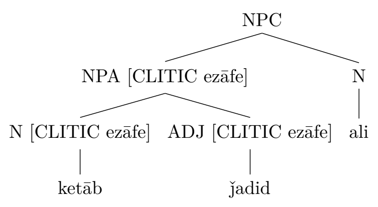
Ezāfe Construction and Noun Complements

The head of a noun phrase is a noun that might take a number of dependents before or after the head noun. The head nouns should be marked with Ezāfe as the phrasal affix (Samvelian, 2007) to be combined with the post-nominal elements, as discussed in Section 2.3. If the next element of the head noun marked with Ezāfe is a modifier, the Head-Adjunct-Schema is used, and if the next element of the head noun marked with Ezāfe is a complement the Head-Complement-Schema is used, as represented in the tree diagram of Example 4.34. In this example, the label 'NPA' is used for combining the head noun 'کتاب' /ketāb/ 'book' and the adjective 'جدید' /ǰadid/ 'new', as its modifier. Moreover, the the label 'NPC' is used for combining the head noun 'کتاب' /ketāb/ 'book' and the proper noun 'علی' /ali/ 'Ali', as its complement to construct a possessive construction.

(4.34) او کتاب جدید علی را خواند.

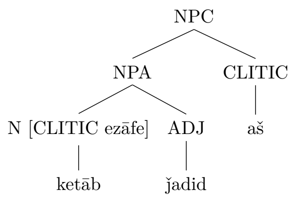
u	ketāb-e	ǰadid-e	ali	rā	xānd
he/she	book-EZ	new-EZ	Ali	DOM	read.3SG

'He/she read Ali's new book.'



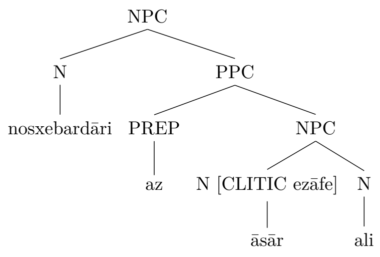
Possessive constructions are expressed by either the Ezāfe construction described in Example 4.34 or possessive clitics. If a clitic is used for possessive construction, the Head-Complement-Schema is used for combining the possessive clitic and the head noun as represented in the tree diagram of Example 4.35.

(4.35) او کتاب جدیدش را خواند.
u ketāb-e ǰadid=aš rā xānd
he/she book-EZ new=3SG DOM read.3SG
'He/she read his/her new book.'



Using Ezāfe as a genitive case marker results in having a noun phrase or a determiner phrase as the complement of the head noun. It is possible for a noun to require a prepositional phrase in its argument structure, like 'نسخه‌برداری' /nosxebardāri/ 'copy' which requires a prepositional phrase headed with the preposition 'از' /az/ 'of, from, than' in Example 4.36:

(4.36) نسخه‌برداری از آثار علی جرم است.
nosxebardāri az āsār-e ali ǰorm ast
copy from work-EZ Ali crime is
'Copying from Ali's works is a crime.'

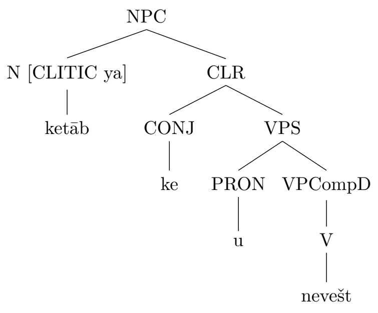


Based on the study of Taghvaipour (2005), the relative clauses are treated as the complement of nouns because of using the restrictive marker 'ی' /-i/ after the noun and the relativizer 'که' /ke/ 'that'. This idea is represented in the tree diagram of Example 4.37. In this example, the noun 'کتاب' /ketāb/ 'book' is marked with the restrictive marker 'ی' /-i/ encoded in its POS tag, and it has a Head-Complement relation with the relative clause. The relative clause, which is labeled 'CLR', is marked with the relativizer 'که' /ke/ 'that' that follows the restrictive marker 'ی' /-i/.

(4.37) کتابی که او نوشت مهم است.

ketāb-i	ke	u	nevešt	mohem	ast
book-REST	that	he/she	wrote.3SG	important	is

'The book that he/she wrote is important.'



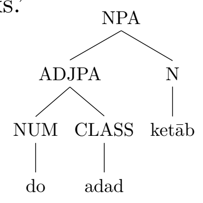
Internal Structure

As mentioned, the internal structure of a noun phrase is highly strict, and it is not possible to change the elements within a noun phrase freely. There are pre-nominal and post-nominal elements in which there exist Head-Adjunct and Head-Complement relations between the head noun and the sister elements. The priority to define the head-dependency relations are on post-nominal elements because most of post-nominal elements have internal structure, whereas this property does not exist on pre-nominal elements (Tabibzadeh, 2012, p. 250). Bateni (1969), Gholamalizadeh (1999), and Megerdoomian (2000) described the type and the position of the pre-nominal elements. Numbers (ordinal (type I) or cardinal), classifiers, superlative adjectives, and titles (type I) come before the head noun, and the Head-Adjunct-Schema is used for them. Determiners and quantifiers are also used before a head noun which are explained in Section 4.4.3.

As shown in Examples 4.34–4.37, the head-dependency relations of the head noun and post-nominals like nouns, prepositional phrases, relative clauses, and reduced relative clauses are realized by applying the Head-Complement-Schema. The Head-Adjunct-Schema is applied for the rest of post-nominal elements.

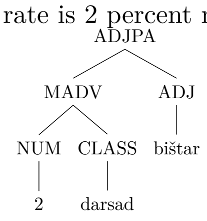
The priority to combine a pre-nominal element to the head is based on its linear appearance. In our annotation, if the sequence of a cardinal number, a classifier, and a noun appears in a sentence, as in Example 4.38, first the number as the head takes the classifier as its modifier, and the Head-Adjunct-Schema is used. Since a number functions as a modifier, the node is labeled 'ADJPA'. This adjectival phrase modifies the head noun, and the label 'NPA' is used by applying the Head-Adjunct-Schema.

(4.38) او دو عدد کتاب دارد.
u do adad ketāb dār-ad
he/she two number book has-3SG
'He/she has two books.'



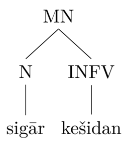
Depending on the local syntactic function, it is possible for the sequence of a number and a classifier to function as an adverb in predicate position as represented in the tree diagram of Example 4.39. For such cases, the label 'MADV' is used for representing a compound adverb. The reason that sequences like '۲ درصد' /do darsad/ '2 percent' in Example 4.39 are recognized as an adverb is that they can be removed and we still have a grammatical sentence. Furthermore, they can be replaced by the adverbs, such as 'بسیار' /besyār/ 'highly' or 'به‌طور ناباورانه' /betore nābāvarāne/ 'unbelievably'.

(4.39) نرخ تورم امسال ۲ درصد بیشتر است.
nerx-e tavarrom-e emsāl 2 darsad bištar ast
rate-EZ inflation-EZ this_year 2 percent more is
'This year's inflation rate is 2 percent more.'



The infinitive form of verbs functions as a noun, regardless of whether the verb is a simple or a light verb. In the annotation, if an infinite light verb and the pre-verbal element form an infinite compound verb that functions as a noun, then the label 'MN' is used for representing a compound noun, as in Example 4.40.

(4.40) سیگار کشیدن برای سلامتی مضر است.
sigār kešidan bara-ye salāmati mozer ast
cigarette pull for-EZ health harmful is
'Smoking is harmful for health.'

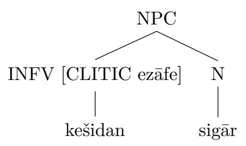


Changing the position of the pre-verbal element changes this structure, as in Example 4.41, because the infinite light verb is marked with Ezāfe and it takes the adjacent noun as its complement.

(4.41) کشیدن سیگار برای سلامتی مضر است.

kešidan-e	sigār	bara-ye	salāmati	mozer	ast
pull-EZ	cigarette	for-EZ	health	harmful	is

‘Smoking is harmful for health.’

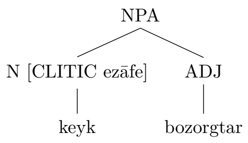


The post-nominal elements of a noun phrase are nouns, simple and comparative adjectives, as in Example 4.42; titles (type II), as in Example 4.43; ordinal numbers (type II), as in Example 4.44; appositions, as in Example 4.45; prepositional phrases, as in Example 4.46; reduced relative clauses, and relative clauses. The last two post-nominal elements are described more in Section 4.5.

(4.42) او کیک بزرگتر را خورد.

u	keyk-e	bozorg-tar	rā	xord
he/she	cake-EZ	big-COMP	DOM	ate.3SG

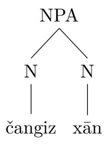
‘He/she ate the bigger cake.’



(4.43) نام او چنگیز خان است.

nām-e	u	čangiz	xān	ast
name-EZ	he/she	Genghis	Khan	is

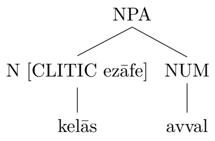
‘His name is Genghis Khan.’



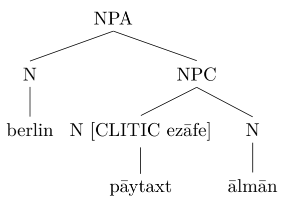
(4.44) او در کلاس اول است.

u	dar	kelās-e	avval	ast
he/she	in	class-EZ	first	is

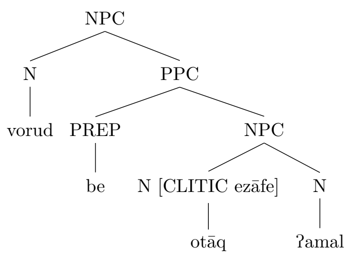
‘He/she is in the first class.’



(4.45) او به برلین پایتخت آلمان سفر کرد.
u be berlin pāytaxt-e ālmān safar kard
he/she to Berlin capital-EZ Germany trip did.3SG
‘He/she travelled to Berlin, the capital of Germany.’



(4.46) ورود به اتاق عمل برای همگان ممنوع است.
vorud be otāq-e ʔamal barā-ye hame-gān mamnuʔ ast
entrance to room-EZ operation for-EZ all-PL forbiden is
‘Entrance to the operation room is forbidden for public.’



4.4.3 Determiner Phrase

There are two types of determiners as quantifiers and demonstratives that are syntactically considered as the head in the Generative Grammar. In the Bijankhan Corpus (Bijankhan, 2004), however, three kinds of determiners are recognized: (a) quantifiers like ‘همه’ /hame/ ‘all’, ‘بعضی’ /baʔzi/ ‘some’, (b) demonstratives like ‘این’ /in/ ‘this’, ‘آن’ /ān/ ‘that’, and (c) interrogative words like ‘چه’ /če/ ‘what’.

Shaghaghi (2002) and Tabibzadeh (2012, p. 250) recognized Persian determiners as pre-nominal dependencies and classified them as adjectives. Tabibzadeh (2012, p. 217) believes that due to the possibility of eliminating quantifiers, as in Example 4.47, they cannot be considered as heads. Aghaei (2006, p. 38) explains that elimination of demonstratives makes sentences ungrammatical, as in Example 4.48.

(4.47) a. همه دوستان او جوان هستند.
hame-ye dust-ān-e u ǰavān hast-and
all-EZ friend-PL-EZ he/she young is-3PL
‘All of his/her friends are young.’

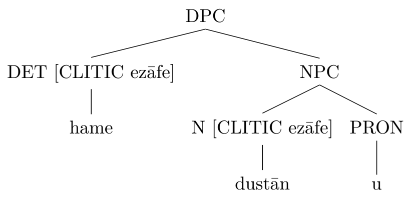
b. دوستان او جوان هستند.
dust-ān-e u ǰavān hast-and
friend-PL-EZ he/she young is-3PL
'His/her friends are young.'

(4.48) a. این اتفاق که حمید ورشکسته شده‌است دروغ است.
in ettefāq ke hamid varšekaste šode-ast doruq ast
this occurrence that Hamid bankrupt become-3SG lie is
'This occurrence that Hamid is bankrupted is a lie.'

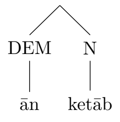
b. * اتفاق که حمید ورشکسته شده‌ست دروغ است.
* ettefāq ke hamid varšekaste šode-ast doruq ast
occurrence that Hamid bankrupt become-3SG lie is

In our annotation, we recognize quantifiers, demonstratives, and interrogatives as heads, as in Examples 4.49–4.51. There are two reasons for that: (a) they agree with verbs in number, and (b) some of them are marked with Ezāfe, which means they require a noun complement. To have a consistent analysis of determiners, whether marked with Ezāfe or not, we recognize them as heads. Furthermore, we do not have a Head-Specifier relation among our defined head-dependency schemas in the manner defined in Pollard and Sag (1994) and described in Section 3.2.3. The potential complements of the determiners are noun phrases, numbers, and prepositional phrases used in partitive constructions, as in Example 4.52.

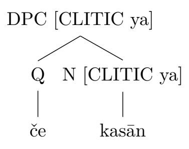
(4.49) همه دوستان او جوان هستند.
hame-ye dust-ān-e u ǰavān hast-and
all-EZ friend-PL-EZ he/she young is-3PL
'All of his/her friends are young.'



(4.50) آن کتاب برای علی مهم است.
ān ketāb barā-ye ali mohem ast
that book for-EZ Ali important is
'That book is important for Ali.'

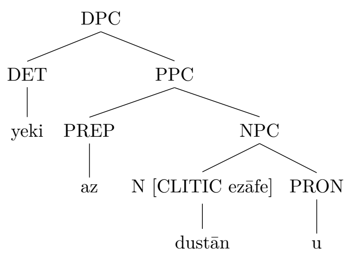


(4.51) چه کسانی دوست او هستند؟
čе kas-ān-i dust-e u hast-and
what person-PL-INDEF friend-EZ he/she is-3PL
'Who are his/her friends?'



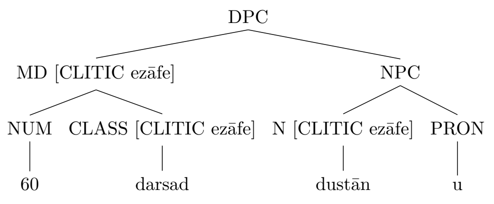
In partitive constructions, the partitive element is recognized as a quantifier and it agrees in number with the verb. Therefore, we also recognize this set of quantifiers as a head. The quantifiers like 'بعضی' /baʔzi/ 'some', 'یکی' /yeki/ 'one', 'تعدادی' /teʔdādi/ 'a number', etc., require a prepositional phrase headed with the preposition 'از' /az/ 'of, from, than' as their complements, as in Example 4.52.

(4.52) یکی از دوستان او جوان است.
yek-i az dust-ān-e u ǰavān ast
one-INDEF from friend-PL-EZ he/she young is
'One of his/her friends is young.'



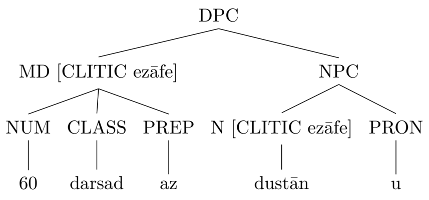
Example 4.39 demonstrated that the sequence of a number and a classifier forms an adverb. Marking this sequence with Ezāfe in Example 4.53 changes its function to a quantifier to form a partitive construction. The reason to consider this sequence as a partitive quantifier is that it can be replaced by a quantifier like 'همه' /hame/ 'all' which is marked by Ezāfe or 'بعضی' /baʔzi/ 'some'.

(4.53) ۶۰ درصد دوستان او جوان هستند.
60 darsad-e dust-ān-e u ǰavān hast-and
60 percent-EZ friend-PL-EZ he/she young is-3PL
'60 percent of his/her friends are young.'



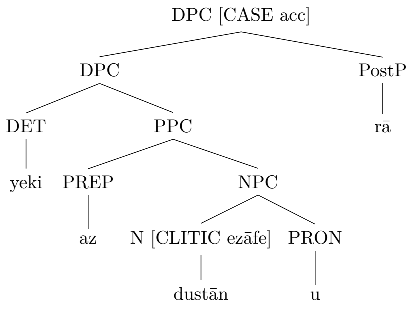
The preposition 'از' /az/ 'of, from, than' in Example 4.54 functions as Ezāfe and not as a preposition. This preposition can be replaced by Ezāfe without any changes on the syntactic construction of the constituent, as in Example 4.53. In our annotation, the sequence of these words is combined together under one node labeled 'MD' as a compound determiner, and it is recognized as the head of the determiner phrase.

(4.54) ۶۰ درصد از دوستان او جوان هستند.
60 darsad az dust-ān-e u ǰavān hast-and
60 percent from friend-PL-EZ he/she young is-3PL
'60 percent of his/her friends are young.'



Any of the mentioned determiner phrases might appear in the direct object position. Similar to noun phrases that the Head-Complement-Schema is used for combing the noun phrase and the particle 'را' /rā/, this schema is used for combining a determiner phrase and the particle 'را' /rā/. The combination is labeled 'DPC', as in Example 4.55.

(4.55) یکی از دوستان او را دیدم.
yek-i az dust-ān-e u rā did-am
one-INDEF from friend-PL-EZ he/she DOM saw-1SG
'I saw one of his/her friends.'



Contrary to Example 4.54, in Example 4.55, the preposition 'از' /az/ 'of, from, than' functions as a preposition and not Ezāfe. The reason is that if the preposition 'از' /az/ 'of, from, than' is replaced by Ezāfe, the sentence will be ungrammatical, as in Example 4.56.

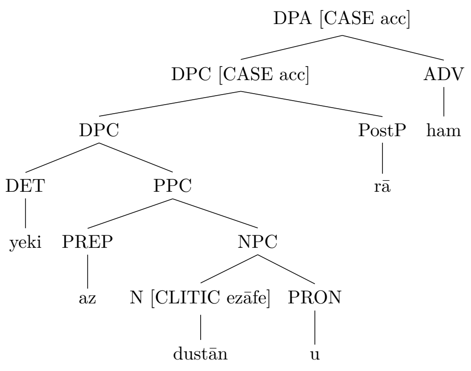
(4.56) * یکی دوستان او را دیدم.
* yek-i-e dust-ān-e u rā did-am
one-INDEF-EZ friend-PL-EZ he/she DOM saw-1SG

There are a number of adverbs, such as ‘نیز’ /niz/ ‘also’, and ‘هم’ /ham/ ‘too’, that modify a determiner phrase, as in Example 4.57.

(4.57) یکی از دوستان او را هم دیدم.

yek-i	az	dust-ān-e	u	rā	ham	did-am
one-INDEF	from	friend-PL-EZ	he/she	DOM	too	saw-1SG

‘I saw one of his/her friends too.’



4.4.4 Adjectival Phrase

The head of an adjectival phrase is an adjective. Potentially, the complement of an adjective can be a determiner phrase, a noun phrase, a number as a date expression, or a prepositional phrase. The number of arguments that adjectives require varies from zero to two. The adjective ‘ثروتمند’ /servatmand/ ‘rich’ requires no argument, and it mostly appears in the predicative position, as in Example 4.58.

(4.58) این مرد ثروتمند است.

in	mard	servatmand	ast
this	man	rich	is

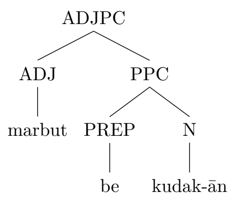
‘This man is rich.’

The required argument for the adjective ‘مربوط’ /marbut/ ‘related’ is one, and it is two for the adjective ‘بازدارنده’ /bāzdārande/ ‘disincentive’ (Tabibzadeh, 2012, pp. 269–270). The adjective ‘مربوط’ /marbut/ ‘related’ requires a prepositional phrase headed with the preposition ‘به’ /be/ ‘to’ as its complement represented in Examples 4.59.

(4.59) مشکلات مربوط به کودکان مهم است.

moškel-āt-e	marbut	be	kudak-ān	mohem	ast
problem-PL-EZ	related	to	child-PL	important	is

‘Problems related to children are important.’

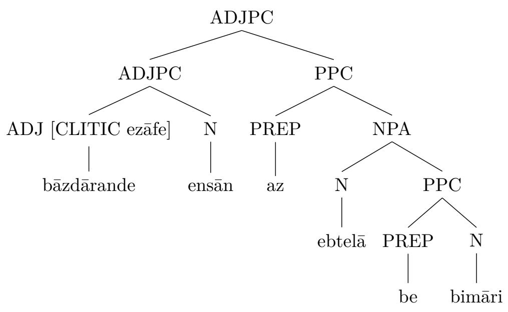


The adjective 'بازدارنده' /bāzdārande/ 'preventive' in Example 4.60 is marked with Ezāfe, and it requires a noun as one of its complement. The second complement is the prepositional phrase headed with the preposition 'از' /az/ 'of, from, than'.

(4.60) واکسیناسیون بازدارنده انسان از ابتلا به بیماری است.

vāksināsiyun	bāzdārande-ye	ensān	az	ebtelā	be	bimāri	ast
vaccination	preventive-EZ	human	from	affection	to	illness	is

'The vaccination prevents a human from getting a disease.'

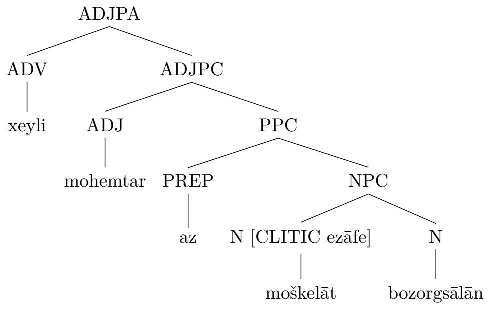


The comparative form of a simple adjective requires a prepositional phrase with the headed preposition 'از' /az/ 'of, from, than' as its complement represented in the tree diagram of Example 4.61. As shown in this example, an adverb can appear before an adjective to modify it. The Head-Adjunct-Scheme is used for combining the adjective and the adverb.

(4.61) مشکلات کودکان خیلی مهمتر از مشکلات بزرگسالان است.

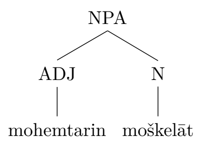
moškel-āt-e	kudak-ān	xeyli	mohem-tar	az
problem-PL-EZ	child-PL	very	important-COMP	from
moškel-āt-e	bozorgsāl-ān	ast		
problem-PL-EZ	adult-PL	is		

'Children's problems are highly more important than adults' problems.'



The superlative form of adjectives functions as a modifier of a noun or the complement of a copula, as in Example 4.62.

(4.62) مهمترین مشکلات مربوط به کودکان است.
mohem-tarin moškel-āt marbut be kudak-ān ast
important-SUP problem-PL related to child-PL is
'The most important problems are related to children.'



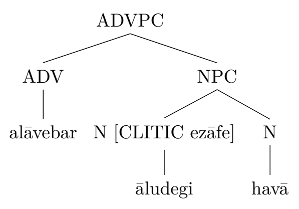
Compound adjectives like 'سنگین وزن' /sangin vazn/ 'heavy-weighted' consist of an adjective like 'سنگین' /sangin/ 'heavy' and a noun like 'وزن' /vazn/ 'weight'. The two elements combine together and they are labeled as 'MADJ' to be treated as a compound adjective.

4.4.5 Adverbial Phrase

An adverb can potentially take an adjectival phrase, a determiner phrase, a prepositional phrase, a noun phrase, or a clause as its argument. In Persian, adverbs can be divided into two groups in terms of their markedness with Ezāfe. Adverbs like 'مانند' /mānand/, 'نظیر' /nazir/, 'مثل' /mesl/ need to be marked by Ezāfe when used in a sentence, while adverbs like 'چون' /čon/ and 'همچون' /hamčon/ do not need to be marked by Ezāfe. All of these adverbs mean 'like'.

In terms of the scope, an adverb can have narrow scope to a local context, and it might require an element in its argument structure, such as 'علاوه‌بر' /alāvebar/ 'in addition to' in Example 4.63.

(4.63) علاوه‌بر آلودگی هوا آلودگی صوتی نیز افزایش یافته است.
alāvebar āludegi-ye havā āludegi-ye souti niz afzāyeš
in_addition_to pollution-EZ air pollution-EZ sound also increase
yāfte ast
found is
'In addition to the air pollution, the sound pollution has also increased.'



Furthermore, an adverb can have wide scope over the whole sentence and it does not require an element as its complement, such as 'متأسفانه' /mote?assefāne/ 'unfortunately' or 'به‌طورکلی' /betorekolli/ 'generally' in Example 4.64.

(4.64) به‌طورکلی مشکلات کودکان خیلی مهم است.

betorekolli	moškel-āt-e	kudak-ān	xeyli	mohem	ast
generally	problem-PL-EZ	child-PL	very	important	is

‘Generally children’s problems are very important.’

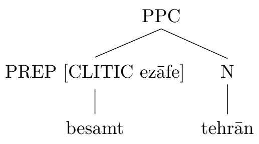
4.4.6 Prepositional Phrase

Prepositions can take a determiner phrase, a number, a noun phrase, or the infinitive form of a verb as a noun in their argument structures. In addition to simple prepositions, AbolhassaniChime (2006) and AbolhasaniChime and Ghayoomi (2006) believe that compound prepositions also exist in Persian. The compound prepositions are the product of incorporating two simple prepositions, or a simple preposition and a noun. In our annotation, compound prepositions like ‘به‌سمت’ /besamte/ ‘to, towards’ are treated as a simple preposition displayed in the tree diagram of Example 4.65.

(4.65) او به‌سمت تهران حرکت کرد.

u	besamt-e	tehrān	harekat	kard
he/she	to-EZ	Tehran	move	did.3SG

‘He/she moved to Tehran.’



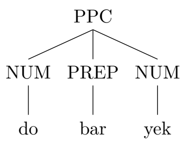
Generally, the number of arguments that prepositions require is one, such as the preposition ‘به‌سمت’ /besamte/ ‘to, towards’ in Example 4.65, but it is possible to have two arguments, such as ‘بر’ /bar/ ‘against’ in Example 4.66.

(4.66) تیم فوتبال آث‌میلان تیم کره را دو بر یک شکست داد.

tim-e	futbāl-e	āsemilān	team-e	kore	rā	do	bar
team-EZ	football-EZ	AC-Milan	team-EZ	Korea	DOM	two	against

yek	šekast	dād
one	beat	gave.3SG

‘AC-Milan football team beat the Korean team two to one.’



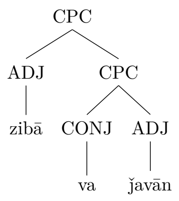
4.4.7 Coordination Phrase

Coordinations, such as ‘و’ /va/ ‘and’, ‘اما’ /ammā/ ‘but’, or ‘یا’ /yā/ ‘or’, are freely used in various syntactic constructions to coordinate two or more words or phrases. It is possible for a comma to function as a coordination as well (Maier and Kübler, 2013). In the annotation of coordination constructions, we recognize coordinators as heads, and their complements can be symmetric or asymmetric lexical items or phrases. Examples 4.67 and 4.68 represent a symmetric coordination phrase and an asymmetric coordination phrase, respectively. In some special syntactic constructions, it is possible for one argument of the coordination phrase to be extraposed. This phenomenon is described in Section 4.7.

(4.67) او دختری زیبا و جوان است.

u	doxtar-i	zibā	va	ǰavān	ast
he/she	girl-INDEF	beautiful	and	young	is

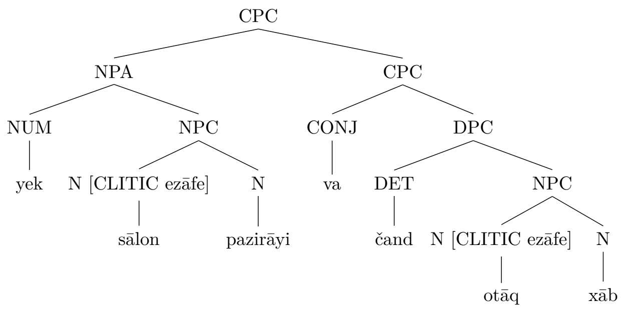
‘She is a beautiful and young girl.’



(4.68) این خانه یک سالن پذیرایی و چند اتاق خواب دارد.

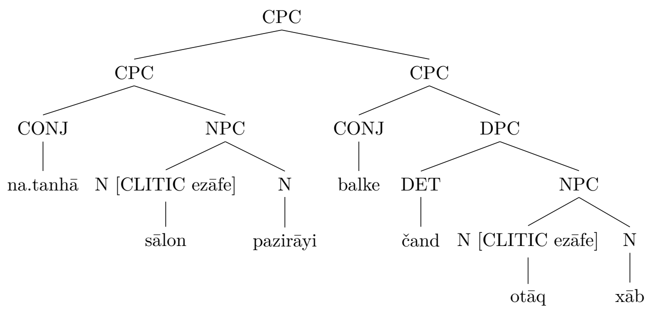
in	xāne	yek	sālon-e	pazirāyi	va	čand	otāq-e	xāb	dārad
this	house	one	hall-EZ	reception	and	several	room-EZ	sleep	has.3SG

‘This house has one reception hall and several sleeping rooms.’



In the annotation of correlative coordinations, each pair of the correlative conjunction is a head and it combines with the sister element through the Head-Complement-Schema. Then, the Head-Complement-Schema is used for combining each pair of the correlative conjunction under one node represented in the tree diagram of Example 4.69.

(4.69) این خانه نه‌تنها سالن پذیرایی بلکه چند اتاق خواب دارد.
in xāne na.tanhā sālon-e pazirāyi balke čand otāq-e xāb
this house not.only hall-EZ reception but several room-EZ sleep
dārad
has.3SG
'This house has not only a reception hall but also several sleeping rooms.'

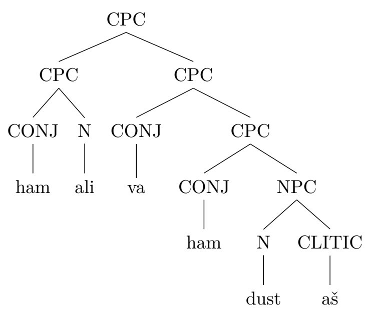


It is possible to use a coordinator between the pairs of correlative conjunctions as well like:

'... هم ... و هم'	/ham ... va ham .../	'both ... and ...'
'... نه ... و نه'	/na ... va na .../	'neither ... nor ...'
'... خواه ... و یا'	/xāh ... va yā .../	'whether ... or ...'
'... چه ... و چه'	/če ... va če .../	'either ... or ...'

In the annotation of these syntactic constructions, we consider each of the elements like 'هم' /ham/, 'نه' /na/, 'خواه' /xāh/, and 'چه' /če/ as a head. To annotate the coordination /va/ 'and', we select this element as the head, and the phrases with correlative conjunctions are the coordination argument as represented in the tree diagram of Example 4.70. It is possible for the left pair of the correlative conjunctions, along with the coordination 'و' /va/ 'and' to be extraposed to the post-verbal position. Because of this property, Tabibzadeh (2012, p. 330) calls these coordinations 'split coordinators'. In Section 4.7, the extraposition of such syntactic constructions is described.

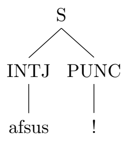
(4.70) هم علی و هم دوستش به تهران رفتند.
ham ali va ham dust=aš be tehrān raft-and
also Ali and also friend=3SG to Tehran went-3PL
'Both Ali and his friend went to Tehran.'



4.4.8 Interjection

Interjections are a class of words that construct a sentence or an utterance without a verb, as in Example 4.71. In these syntactic constructions, the interjector is the head that might take a clause as its complement (Tabibzadeh, 2012, p. 336), as in Example 4.72.

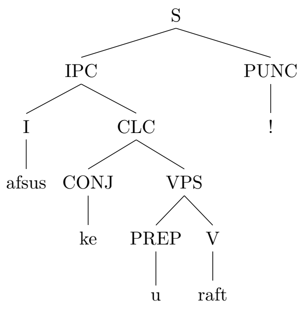
(4.71) افسوس!
afsus
pity
'What a pity!'



(4.72) افسوس که او رفت!

afsus	ke	u	raft
pity	that	he/she	went.3SG

'What a pity that he/she went!'



4.5 Clauses

4.5.1 Relative Clause

The annotation scheme of the relative clause is described in Sections 4.4.2 with the relevant examples. Therefore, we do not repeat it here.

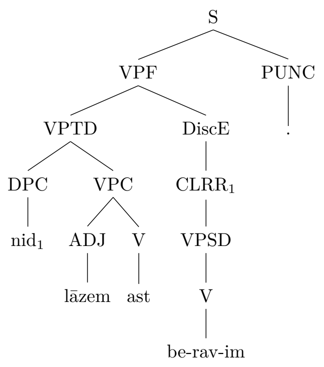
4.5.2 Reduced Relative Clause

A reduced relative clause is a relative clause that has no relativiser. This syntactic construction is labeled 'CLRR' rather than 'CLR', as represented in the tree diagram of Example 4.73. The phenomenon demonstrated in this example is similar to Example 4.31 on page 91. The only difference is that in Example 4.31 a relative clause is used with the relativizer 'که' /ke/ 'that', but in Example 4.73 no relativizer is used.

(4.73) لازم است برویم.

lāzem	ast	be-rav-im
necessary	is	SUBJ-go-1PL

'It is necessary [that] we go.'



4.5.3 Free Relative Clause

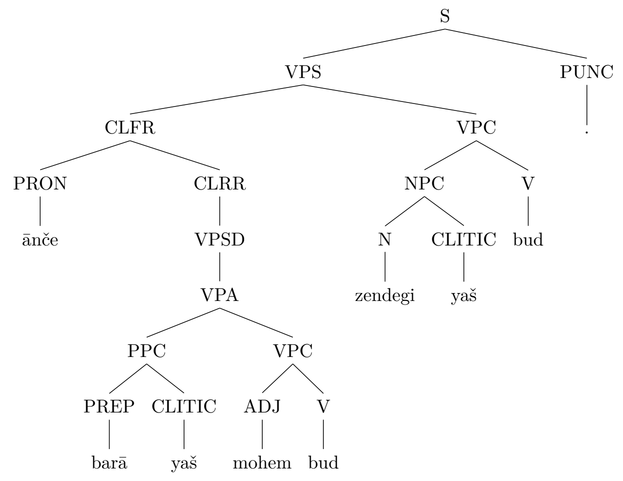
In the syntactic construction of a free relative clause in Persian, there is no explicit antecedent, and the clause might take the place of an argument in the matrix clause. Taghvaipour (2005) expressed that most of the clauses that start with 'هرچه' /harče/ 'whatever', 'هرکه' /harke/ 'whoever', 'هروقت' /harvaqt/ 'whenever', etc., and take a noun phrase position in the matrix clause create a free relative clause. Taghvaipour further added that free relative clauses might be marked by the optional relativiser 'که' /ke/ 'that'. In all of the examples provided here, the free relative clauses function as one of the arguments of the matrix clause.

In our annotation, we use the label 'CLFR' for free relative clause constructions in subject and object positions, as in Examples 4.74 and 4.75 , respectively. If the relativizer does not exist, the embedded clause is treated as a reduced relative clause and the clause is labeled 'CLRR', as in Examples 4.74.

(4.74) آنچه برایش مهم بود زندگیش بود.

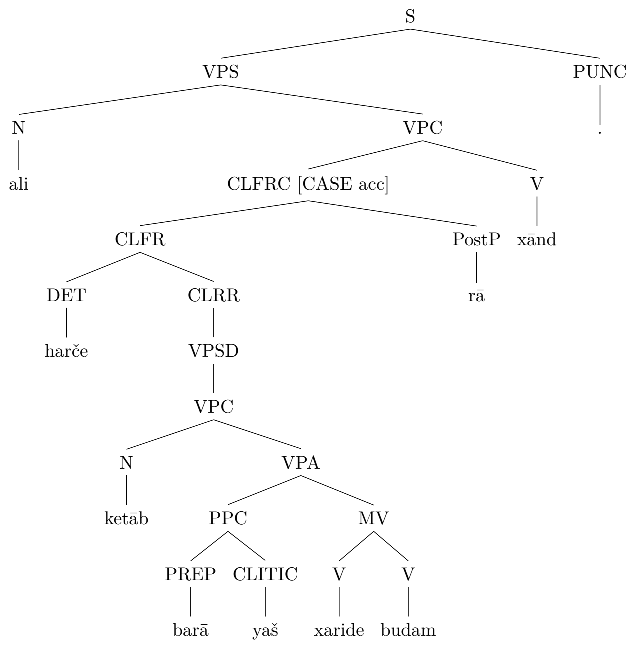
ānče	barā=yaš	mohem	bud	zendegi=yaš	bud
whatever	for=3SG	important	was.3SG	life=3SG	was.3SG

'What was important for him/her was his/her life.'



In Example 4.75, the particle 'را' /rā/ after the free relative clause functions as a specificity marker (Karimi, 1999, p. 13). In our annotation, a Head-Complement relation is defined between the free relative clause and the specificity marker, and we use the label 'CLFRC' as represented in the tree diagram of the Example 4.75.

(4.75) علی هرچه کتاب برایش خریده بودم را خواند.
ali harče ketāb barā=yaš xaride bud-am rā xānd
Ali whatever book for=3SG bought was-1SG DOM read.3SG
'Ali read whatever book I had bought for him/her.'



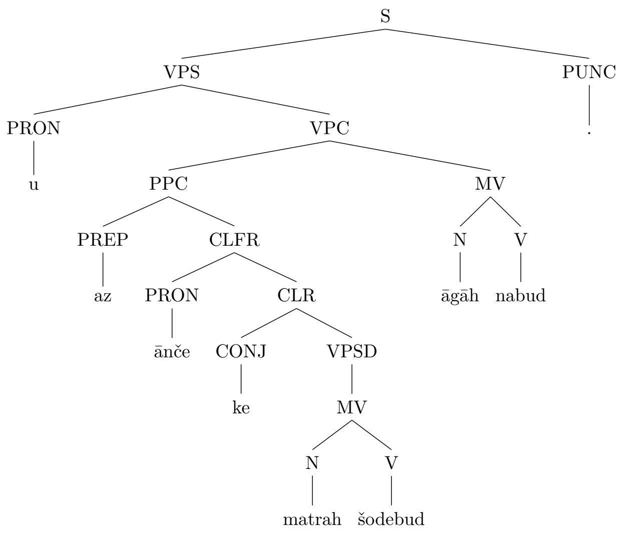
If a relativizer is used in a free relative clause, the embedded clause is treated as a relative clause and labeled 'CLR'. Then, it combines with the elements like 'هرچه' /harče/ 'whatever' and 'هرکه' /harke/ 'whoever', and the label

‘CLFR’ is assigned, as in Example 4.76. In this example, the free relative clause ‘آنچه که مطرح شده‌بود’ /ānče ke matrah šode bud/ ‘what had been mentioned’ is used as the complement of the preposition ‘از’ /az/ ‘of, from, than’ in the object position of the verb ‘آگاه بودن’ /āgāh budan/ ‘be aware’.

(4.76) او از آنچه که مطرح شده‌بود آگاه نبود.

u	az	ānče	ke	matrah	šode_bud	āgāh
he/she	from	whatever	that	mention	become_was.3SG	aware

na-bud
NEG-was.3SG
‘He/she was not aware of what had been mentioned.’



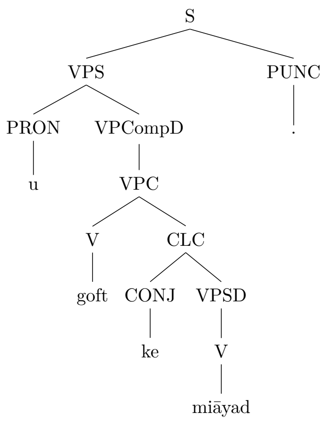
4.5.4 Complement Clause

Phrasal complements are marked with the complementizer ‘که’ /ke/ ‘that’ in Persian. They are represented with the label ‘CLC’ in the tree diagram of Example 4.77. In this example, the verb ‘گفتن’ /goftan/ ‘say, tell’ requires a subject and two complements in its argument structure. If one of its arguments, which functions as an object, comes after the verb, this argument is marked with the complementizer ‘که’ /ke/ ‘that’. A prepositional phrase, which is the second complement of this verb, is dropped in this example, therefore it is represented as ‘VPCompD’.

(4.77) او گفت که می‌آید.

u	goft	ke	mi-ā-yad
he/sh	said.3SG	that	IMPF-come-3SG

‘He/she said [to me] that he/she comes.’

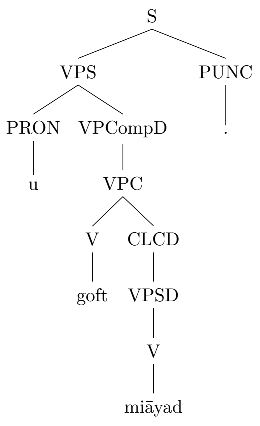


4.5.5 Complement Clause without a Complementizer

It is possible for a phrasal complement belonging to the argument structure of the verb in the matrix clause to be used without the complementizer 'که' /ke/ 'that'. To be consistent with clausal complements, we use the label 'CLCD' to represent that the complementizer is dropped, as in Example 4.78.

(4.78) او گفت می‌آید.

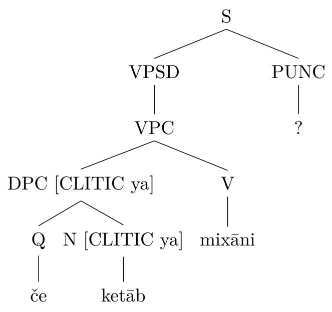
u goft mi-ā-yad
he/sh said.3SG IMPF-come-3SG
'He/she said [to me that] he/she comes.'



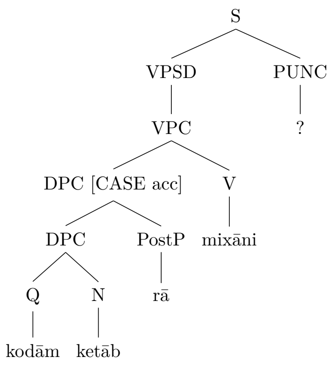
4.5.6 Interrogative Clause

The interrogative words, like 'چه' /če/ 'what' and 'کدام' /kodām/ 'which', are recognized as determiners. These interrogative words take a noun as their arguments. Since in these syntactic constructions, the interrogative phrase remains in-situ, we annotate them in a manner similar to declarative sentences, as in Examples 4.79 and 4.80.

(4.79) چه کتابی می‌خوانی؟
če ketāb-i mi-xān-i
what book-INDEF IMPF-read-2SG
'What book do you read?'

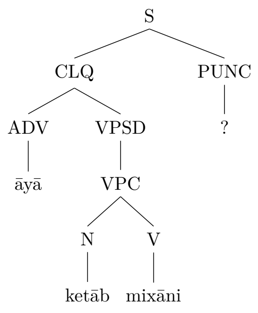


(4.80) کدام کتاب را می‌خوانی؟
kodām ketāb rā mi-xān-i
which book DOM IMPF-read-2SG
'Which book do you read?'



To annotate polar interrogative sentences, we use the label 'CLQ', as in Example 4.81.

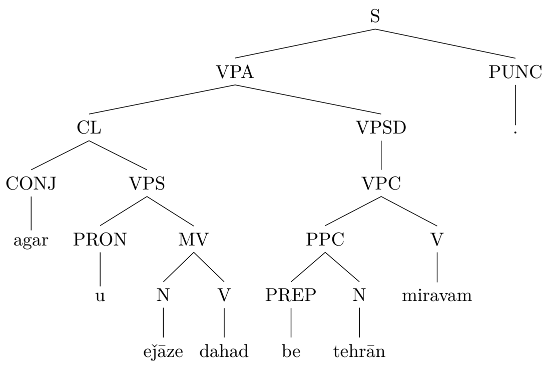
(4.81) آیا کتاب می‌خوانی؟
āyā ketāb mi-xān-i
whether book IMPF-read-1SG
'Do you read a book?'



4.5.7 Other Types of Clauses

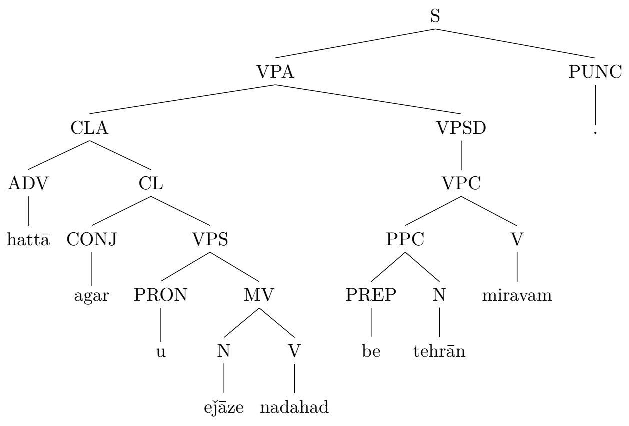
There are other types of clauses which are excluded from the clauses described above, such as subordinate or conditional clauses. Since these syntactic constructions share properties, we use the label 'CL' for them, as in Example 4.82. The Head-Adjunct relation is used for combining the subordinate or conditional clauses and the matrix clause.

(4.82) اگر او اجازه دهد به تهران می‌روم.
agar u ejāze dah-ad be tehrān mi-rav-am
if he/she permission give-3SG to Tehran IMPF-go-1SG
'If he/she gives permission, I will go to Tehran.'



If an adverb has wide scope and modifies a subordinate clause, like the focus adverb 'حتی' /hattā/ 'even', the Head-Adjunct relation is used for combining the adverb and the subordinate clause, and they are labeled 'CLA', as in Example 4.83.

(4.83) حتی اگر او اجازه ندهد به تهران می‌روم.
hattā agar u eǰāze na-dah-ad be tehrān
even if he/she permission NEG-give-3SG to Tehran
mi-rav-am
IMPF-go-1SG
'Even if he/she does not give permission, I will go to Tehran.'



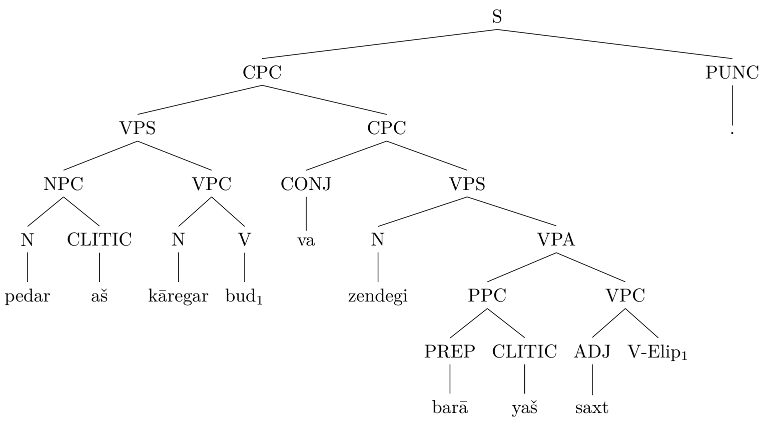
4.6 Ellipsis

In elliptical constructions, one or more words that are required in the sentence are omitted. Depending on the type of ellipsis, the elliptical constructions are created. In our annotation, ellipsis is represented by inserting an empty node labeled according to the type of the elliptic element, as in Example 4.84. In this example, the elliptic element is a verb displayed by the label ‘V-Elip’ in the tree analysis. Moreover, the ellipsis and the retrieved element are co-indexed to share the argument structure.

(4.84) پدرش کارگر بود و زندگی برایش سخت.

pedar=aš	kāregar	bud	va	zendegi	barā=yaš	saxt
father=3SG	worker	was.3SG	and	life	for=3SG	hard

‘His/her father was a worker and life [was] hard for him/her.’



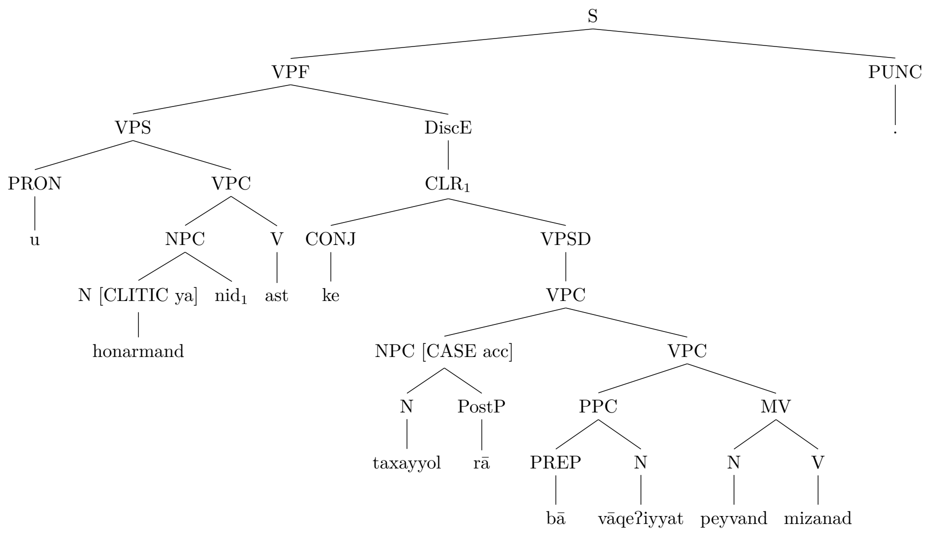
4.7 Discontinuity

At the sentence level, it is possible for a phrase or a clause, which belongs to the argument structure of a head, to extrapose. After applying the Head-Subject-Schema to realize the subject of a sentence, the extraposed element should be bound. As mentioned, we use a trace-based analysis in our annotation such that both the trace and the extraposed element are co-indexed. To bind the extraposed elements, the Head-Filler-Schema is used. To make the trace and the extraposed elements explicit, the empty node ‘*nid*’ and the unary branching node labeled ‘*DiscE*’ are used, as in Example 4.85 where the relative clause is extraposed to the post-verbal position.

(4.85) او هنرمندی است که تخیل را با واقعیت پیوند می‌زند.

u	honarmand-i	ast	ke	taxayyol	rā	ba	vāqeʔiyyat
he/she	artist-REST	is	that	imagination	DOM	with	reality
peyvand	mi-zan-ad						
link	IMPF-hit-3SG						

‘He/she is an artist who links imagination to reality.’

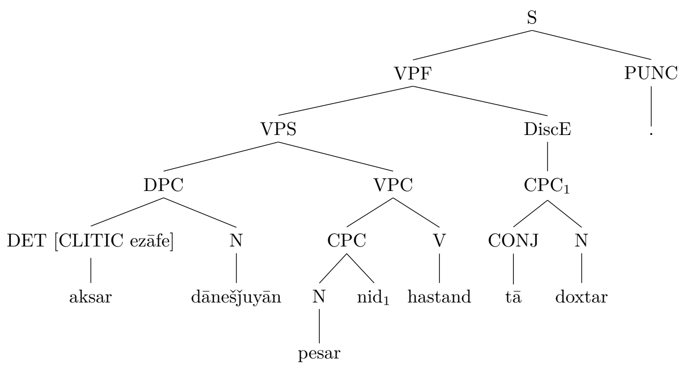


In comparative constructions, we analyze sentences according to the linear appearance of the words. It is possible for comparative constructions to extrapose to the post-verbal position. Since we do not use feature structures, we do not define a new schema for comparative constructions. Therefore, in these constructions, we assume that the elements headed by the coordination 'تا' /tā/ 'rather than' are extraposed to the post-verbal position, as in Example 4.86. These elements are bound by the Head-Filler-Schema.

(4.86) اکثر دانشجویان پسر هستند تا دختر.

aksar-e	dānešǰu-yān	pesar	hast-and	tā	doxtar
majority-EZ	student-PL	boy	is-3PL	rather_than	girl

'The majority of students are boys rather than girls.'

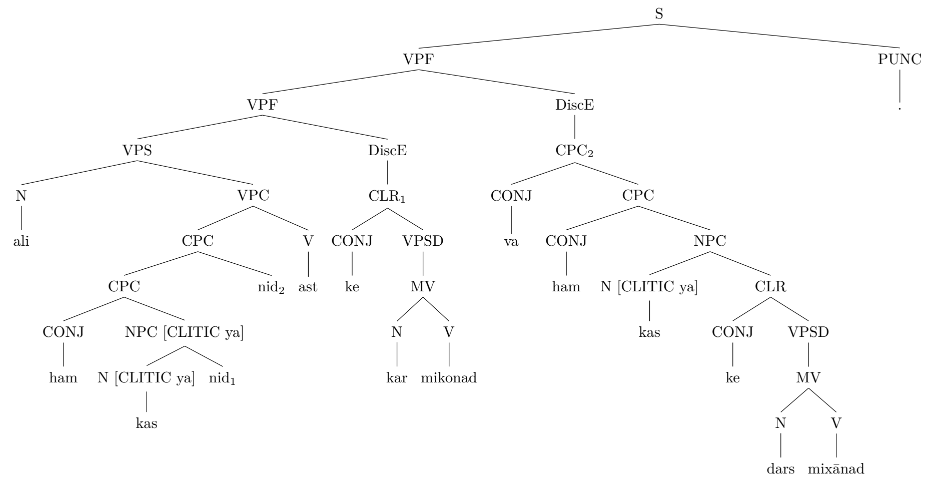


There is also a possibility that in coordination phrases, which are constructed by correlative conjunctions, one of the elements extraposes to the post-verbal position, as in Example 4.87. In this example, there are two instances of extrapositions: (a) the extraposition of the relative clause to a post-verbal position, and (b) the extraposition of the correlative conjunction. Both extrapositions are bound by the the Head-Filler-Schema.

(4.87) علی هم کسی است که کار می‌کند و هم کسی که درس می‌خواند.

ali ham kas-i ast ke kār mi-kon-ad va ham
Ali also person-REST is that work IMPF-do-3SG and also
kas-i ke dars mi-xān-ad
person-REST that lesson IMPF-read-3SG

'Ali is both a person who works and a person who studies.'



4.8 Summary

In this chapter, we described the annotation scheme used for developing our HPSG-based treebank. The annotation was restricted to the lexical, phrasal, and clausal elements. Furthermore, various constructions were described along with examples and their corresponding tree analyses.

Part II

Computational Approaches

Chapter 5

Bootstrapping the Persian Treebanking

5.1 Introduction

In previous chapters, the cruciality of developing and making annotated data available for feeding linguistic investigation and data driven approaches in human language technologies were discussed. It was pointed out that while a great deal of effort has been expended to develop annotated data, such as treebanks, for languages like English and German, similar data sources are not available for all languages. There are many languages like Persian that have been given less attention, and consequently less annotated data is available for them. In this chapter, we describe a bootstrapping approach for bridging the gap to develop a treebank for Persian. When this research was commenced, there were as of yet no publicly available treebanks for Persian, no statistical parser to process the sentences of this language automatically, and no gold data for evaluation. These shortcomings motivated us to develop the treebank from scratch for Persian using a bootstrapping approach and to make the annotated data publicly available for free.

Bootstrapping is a sample selection approach which aims at increasing the size of the annotated data iteratively and making a steady improvement on the performance as a result; i. e. in each iteration a small fragment of the data and not the whole data is annotated, and incrementally the data is annotated and added to the annotated data set. This approach is used for many data oriented NLP applications, like parsing (Aldezabal et al., 2000), machine translation (Pal and Bandyopadhyay, 2012), information extraction (Iosif et al., 2012), named-entity recognition (Teixeira et al., 2011), word sense disambiguation (Diab, 2004), term extraction (Heid, 2000), opinion extraction (Xia et al., 2009), topic analysis (Ferret and Grau, 2002), lexicon learning (Thelen and Riloff, 2002), and speech synthesis (Bulyko and Ostendorf, 2002). Moreover, this method is employed in the development of monolingual treebanks (Huang et al., 2000; Brants et al., 2002; Nivre and Megyesi, 2007; Dridan and Baldwin, 2010; Seraji et al., 2012a; Rasooli et al., 2013) as well as parallel treebanks (Volk and Samuelsson, 2004).

This chapter contains six sections. In Section 5.2, the tool employed for our data annotation is introduced. Next, the algorithm used for bootstrapping the development of the treebank is explained in Section 5.3. In the bootstrapping process, the most frequent grammar rules are extracted from the treebank, and they are defined in the annotation tool to further annotate unseen data. The annotation process is done in four steps, which are explained in Section 5.4. In Section 5.5, the amount of human intervention needed to annotate the data and the accuracy of the defined grammar rules are evaluated. Section 5.6 summarizes this chapter.

The content of this chapter is mainly based on these two papers:

- Ghayoomi, Masood (2012) "Bootstrapping the development of an HPSG-based treebank for Persian". In *Linguistic Issues in Language Technology*, 7 (1). CSLI Publications.

- Ghayoomi, Masood (2012) "From grammar rule extraction to treebanking: A bootstrapping approach". In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, May, 23–25, 2012; Istanbul, Turkey, pp: 1912–1919.

5.2 The CLaRK System

To start treebanking, a tool for the data annotation is required. To this end, we employ the CLaRK system[1] (Simov and Osenova, 2002) for our application. The CLaRK system is an XML[2]-based tool for corpus development implemented in Java. The development of the system is carried out by the Tübingen-Sofia International Graduate Program. Minimization of human intervention during the creation of the language resource has been the main aim for designing this tool. This tool is primarily used for developing a treebank for Bulgarian based on HPSG, called the BulTreeBank (Simov et al., 2001).

There is a Unicode-based tokenizer in the system to support the tokens. Additionally, there is a deterministic finite state automaton behind the system

[1]`http://www.bultreebank.org/clark/`
[2]eXtensible Markup Language

to support writing cascaded finite state grammar rules and facilitate the search for patterns defined as regular expressions. It is possible to write the grammar rules as regular expressions in the system to facilitate data annotation. The defined regular expressions can be applied at the data bidirectionally, either left-to-right or right-to-left. The XML Path Language (XPath) is the query language used in the system to support navigation over the whole document and to select specific elements. To apply regular expressions at sentences in an XML document, first the patterns as regular expressions are converted into automata, then into the XPath query language. The system is empowered with XSLT[3], which is a language for transforming XML documents to other XML documents and it is used as part of XSL[4], the stylesheet language for XML. Supporting XSLT makes it possible to draw graphical tree structures from the XML document. Figure 5.1 illustrates the tree analysis of Example 5.1 created by the CLaRK system.

(5.1) بورن به‌اتفاق یکی از اقوامش به شهر پراگ مهاجرت کرد.

born	beʔettefāq-e	yek-i	az	aqvām=aš	be	šahr-e	perāg
Born	with-EZ	one-INDEF	from	relatives=3SG	to	city-EZ	Prague

mohājerat	kard
move	did.3SG

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

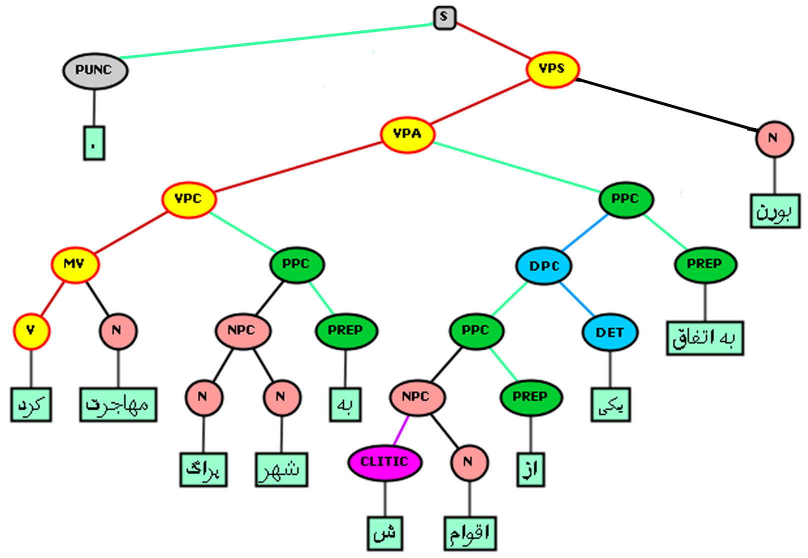


Figure 5.1: Tree analysis of Example 5.1 created by CLaRK

The entire XML document is controlled with DTDs. DTDs play several roles in the CLaRK system (Simov and Osenova, 2003, 2002; Osenova and Simov,

[3]eXtensible Stylesheet Language Transformations
[4]eXtensible Stylesheet Language

- sign
  - **WORD**:*word*
    - PHONE:*phone-list*
    - SYNSEM:*synsem*
      - LOCAL:*local*
        - CATEGORY:*category*
          - HEAD:*head*
        - CONTENT:*content*
          - INDEX:*index*
        - CONTEXT:*context*
      - NON-LOCAL:*non-local*
  - **PHRASE**:*phrase*
    - DAUGHTERS:*constituent structures*
      - HEAD-DTR:*sign*
      - COMP-DTRS:*list of phrases*
        - HEAD-ADJUNCT:*head-adjunct*
          - HEAD-DTR:*word*
          - ADJUNCT-DTR:*phrase*
        - HEAD-COMP:*head-complement*
          - HEAD-DTR:*word*
          - COMP-DTR:*phrase*
        - HEAD-SUBJ:*head-subject*
          - HEAD-DTR:*word*
          - SUBJECT-DTR:*phrase*
        - HEAD-FILLER:*head-filler*
          - HEAD-DTR:*word*
          - FILLER-DTR:*phrase*
    - NON-HEADED-PHRASE:*non-headed-phrase*

.

Figure 5.2: Sample of HPSG signature type hierarchy

2004): (a) they represent a sort hierarchy similar to the sort hierarchy used in TRALE (Penn, 2004) as represented in Figures 5.2 and 5.3, (b) they function as a constraint on ID schemas and linear precedence in HPSG, and (c) they monitor and check the consistency during the annotation process.

The CLaRK system supports the cascaded regular grammar introduced by Abney (1996); i. e. the product of one grammar rule is the input to another grammar rule, and the grammar rules defined as regular expressions in the system will recognize only a portion of a string and not the whole string. 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. King and Simov (1998), Simov (2001), and Simov (2002) proved that a finite theory defined as a set of feature graphs is suitable for the representation of HPSG. As a result, the annotated sentences based on this assumption will not have feature structures similar to the normal HPSG, but the analyses are

```
<!DOCTYPE sign [
<!ELEMENT sign (WORD,PHRASE)>
<!ELEMENT WORD (PHONE,SYNSEM)>
<!ELEMENT PHONE (#PCDATA)>
<!ELEMENT SYNSEM (LOCAL,NON-LOCAL)>
<!ELEMENT LOCAL (CATEGORY,CONTENT,CONTEXT)>
<!ELEMENT CATEGORY (HEAD)>
<!ELEMENT HEAD (#PCDATA)>
<!ELEMENT CONTENT (INDEX)>
<!ELEMENT INDEX (#PCDATA)>
<!ELEMENT CONTEXT (#PCDATA)>
<!ELEMENT NON-LOCAL (#PCDATA)>
<!ELEMENT PHRASE (DAUGHTERS,NO-HEADED-PHRASE)>
<!ELEMENT DAUGHTERS (HEAD-DTR,COMP-DTR)>
<!ELEMENT HEAD-DTR (#PCDATA)>
<!ELEMENT COMP-DTR (HEAD-AJCT,HEAD-COMP,HEAD-SUBJ,HEAD-FILLER)>
<!ELEMENT HEAD-AJCT (HEAD-DTR,AJCT-DTR)>
<!ELEMENT HEAD-DTR (WORD)>
<!ELEMENT AJCT-DTR (PHRASE)>
<!ELEMENT HEAD-COMP (HEAD-DTR,COMP-DTR)>
<!ELEMENT COMP-DTR (PHRASE)>
<!ELEMENT HEAD-SUBJ (HEAD-DTR,SUBJ-DTR)>
<!ELEMENT SUBJ-DTR (PHRASE)>
<!ELEMENT HEAD-FILLER (HEAD-DTR,FILLER-DTR)>
<!ELEMENT FILLER-DTR (PHRASE)>
<!ELEMENT NON-LOCAL (NO-HEADED-PHRASE)>
<!ELEMENT NO-HEADED-PHRASE (WORD)>
] >
```

Figure 5.3: Sample of signature type hierarchy based on a DTD format

composed of phrase structure trees enriched with the simulation of basic properties of HPSG, such as morpho-syntactic information to represent the lexical knowledge, structure sharing, sort hierarchy, ID schemas (Head-Subject, Head-Complement, or Head-Adjunct), and binding the slashed elements through the Head-Filler-Schema.

There are several reasons that the CLaRK system is selected as a tool for our application: (a) since there was no accessible treebank for Persian at the time of this research, the data annotation process had to be started from scratch. As a result, a tool is required to provide an environment to define regular grammar rules to speed up the annotation task and to ease and reduce the amount of human effort; (b) since the grammar formalism used in the BulTreeBank (Simov et al., 2003) is similar to our treebank, it was a strong motivation for us to use this tool for our task; (c) the system supports Unicode which nicely handles the Persian script; (d) the data structure of the system's output is an XML document that makes it possible to add layers for representing the linguistic knowledge, such as POS tags, named-entity tags, lemmas, and verb stems.

5.3 Bootstrapping via Grammar Rule Extraction

Stochastic modeling is frequently used in supervised grammar induction. Grammar induction, also known as grammar inference, automata induction, or automatic language acquisition in the literature (Parekh and Honavar, 2000), means to automatically induce the formal grammar of a language from a set of annotated data, such as a treebank. Machine learning approaches can be used for learning the grammar from a treebank to recognize the syntactic patterns.

As said in Section 5.2, the CLaRK system does not support stochastic modeling for the cascaded regular grammars, and defining the regular grammar rules and their hierarchical ordering are done manually. Nevertheless, the reasons mentioned in the previous section persuaded us to use this system to develop a treebank for Persian. To simplify the annotation task, to decrease human intervention in the development of the data source, and to profit more from the CLaRK system, we introduce a non-stochastic grammar induction approach to learn the grammar rules from the annotated Persian data to annotate further unseen data.

In our view, grammar induction is a task that recognizes the grammar G of the sentence S in the 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. After the compilation of the grammar rules R in CLaRK, each rule r in the system is transformed to a deterministic automaton automatically. When an unseen sentence S' is given to the system, given that the local constrains defined for r are satisfied, the grammar rule r is applicable to S'. The result is a robust analysis for strings that satisfy the local constrains.

To find and define the most frequent grammar rules as regular expressions in CLaRK, we establish a bootstrapping process introduced in Algorithm 1 to speed up treebanking.

In this bootstrapping process, 1,000 sentences from the Bijankhan Corpus (Bijankhan, 2004) are pre-processed to realize multi tokens. Next, to provide the seed grammar rules (R_s) and start the annotation process, frequent bigrams, which construct constituents, are defined as regular expressions in CLaRK. Using bigrams to define seed grammar rules is described in Section 5.4.3. The result of applying R_s is shallowly processed sentences. These rules are used for annotating the first 50 sentences of the data set. The result of these applied

Algorithm 1 Bootstrapping process for treebanking

```
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 $K$ most frequent grammar rules from $R_m$ which have met prerequisite
  grammar rules
  Define the $K$ selected grammar rules as regular expressions in CLaRK
  Augment $R_s$ with the $K$ selected grammar rules and remove them from $R_m$
end while
```

grammar rules are checked so as to not over-generate. In the case of over-generation, constraints are defined for them to limit their domain to the local context.

After defining the seed grammar rules, the bootstrapping process is initialized. To this end, the remaining sentences (950 sentences) are segmented to sets containing N sentences (N=*10*). In each iteration, the N sentences are annotated automatically with the seed grammar rules R_s. The shallowly processed sentences are manually annotated to have the complete analyses of the N sentences. In the next step, the manual grammar rules (R_m) are extracted from the completely annotated sentences. The extracted grammar rules are sorted in descending order based on their frequency. For the next step of the bootstrapping process, the K most frequent grammar rules (K=*5*) which have a high degree of reusability and have met their prerequisite grammar rules are selected and defined as new regular expressions in CLaRK. R_s is augmented with the newly selected grammar rules from R_m. Finally, the updated R_s is used for the next iteration. The bootstrapping process continues iteratively, and it terminates when the remaining 950 sentences are annotated completely.

The proposed algorithm for treebanking, in general, might seem very similar to the bootstrapping approach utilized in the development of the German TIGER treebank (Brants et al., 2002) described in Section 3.3.1. But the difference between our approach and their approach is that in our task treebanking is started from scratch without any preliminary annotated data for Persian. Moreover, our data is annotated based on the HPSG formalism.

In the grammar rule extraction step of our proposed model, in each iteration, all the grammar rules are extracted from the set of annotated sentences. Then, the extracted grammar rules are distinguished based on whether they are already defined in the system, or the annotator inserted the grammar rules manually. To make this distinction explicit, the attribute '*r*' is added to the grammar rules defined as regular expressions in CLaRK. The absence of the attribute '*r*' means that the rule is applied manually. Now, there exist two sets of grammar rules: (a) a set of grammar rules which is already defined in CLaRK as regular expressions and does not require redefinition, and (b) a set of grammar rules added in the manual annotation step. The grammar rules can be the potential candidates for selection and definition as new regular expressions in the system to annotate new sentences. To select among the grammar rules that are inserted during the manual annotation step, this question may raise: "Which of the extracted grammar rules have a priority over other grammar rules to be defined as regular expressions in the CLaRK system?"

In the cascaded regular grammar, the product of one rule is the input of another rule, and in each rule only a portion of a string and not the whole string is recognized. An important property of the cascaded regular grammar is that there is an ordering on 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 their hierarchical ordering are also bottom-up; i. e. the grammar rules belonging to the higher level of the grammar rule hierarchy are extracted and defined in CLaRK under the condition that the grammar rules belonging to the lower level of the grammar rule hierarchy are already extracted and defined in the system. As a result, the grammar rules in the lower level of the grammar rule hierarchy have priority over the grammar rules in the higher level of the grammar rule hierarchy. To identify the hierarchy of grammar rules in our algorithm, we create a matrix of $N \times N$ where N is the total number of grammar rules. Each row of this matrix stores the prerequisites of the corresponding rules. Prerequisites are the grammar rules that belong to the lower level of the grammar rule hierarchy. In each iteration, the grammar rules that have already met their prerequisites can be considered as candidates. These candidates should then be ranked to find the most frequent grammar rules inserted by a human annotator and to assign them a priority based on their frequency. Figure 5.4 on the following page illustrates this idea for Example 5.1 on page 121.

As mentioned, after extracting the grammar rules from the tree structures, they are ordered hierarchically. As shown in Figure 5.4, the extracted grammar rules from the tree structures are organized in several hierarchical levels in the bottom-up model. In the first level of the grammar rule hierarchy, the grammar rules are linked to the lexical items shown with a solid-line triangle in the figure. Most of the grammar rules at this level are the ones added to the system in the initialization step to handle multi-tokens and frequent constituents. The

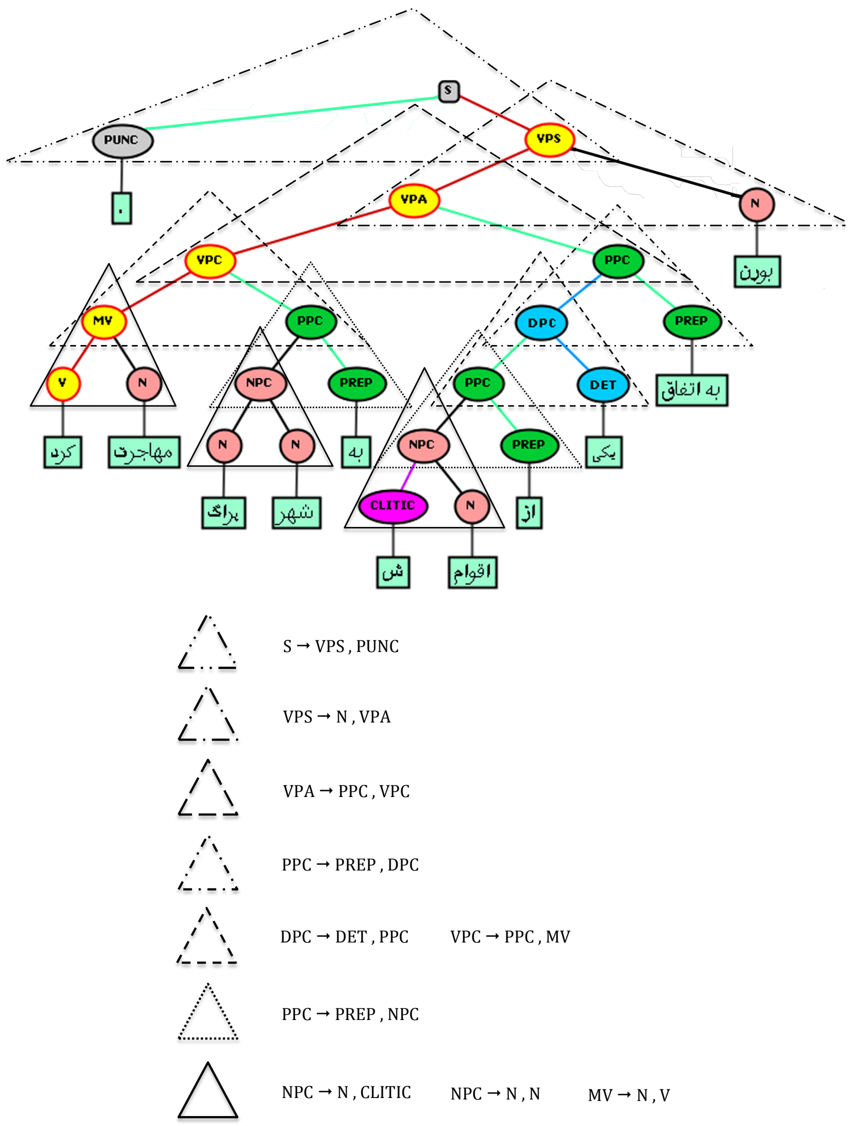


Figure 5.4: Extracting grammar rules and ordering them according to their hierarchical levels

grammar rules of each level in the hierarchy are the ones in which all of their prerequisites in the lower level(s) have been met. Building this hierarchical structure of grammar rules continues in a loop until a grammar rule is met that contains the non-terminal S in its left-hand side. Since not all grammar rules are equally effective, we do not treat the grammar rules in each level equally. As a result, the grammar rules which are very productive and frequent have priority over less productive and infrequent grammar rules to be defined in the system. Based on this idea, the most frequent grammar rules from the ranked list which have satisfied all prerequisites are defined in CLaRK.

5.4 Steps of Treebanking

5.4.1 Pre-processing Step

We found two shortcomings in the original format of the POS tags in the Bijankhan Corpus (Bijankhan, 2004) used in our research. As can be seen in Table 5.1, the length of the tags and the position that certain information is declared are not fixed. To solve the problems, the original tags are converted into the MulText-East[5] framework to encode the morpho-syntactic and semantic information as a single tag. Appendix C provides the POS tags of the Bijankhan Corpus according to the format in the MulText-East framework. In this new tag format, the length of a tag with respect to the main syntactic category is fixed, and each position in the tag corresponds to one specific feature of the word. If the value of certain information is unspecified, then the symbol '−' is used. The lexical features in this linear format can also be displayed in an AVM used in normal HPSG. Figure 5.5 on the following page illustrates the feature structure of the word 'چک' /ček, čak/ 'cheque, whack' with the POS tag 'Ncsp---'.

Table 5.1: Conversion of POS tags in the Bijankhan Corpus into MulText-East framework

Persian word	Transliteration	Meaning	Original tag	Converted tag
چک	ček/čak	cheque;check/whack	N,COM,SING	Ncsp---
چک	ček.e/čak.e	cheque/whack	N,COM,SING,EZ	Ncsp--z
چک	ček	Czech	N,PR,SING	Nasp---
چک	ček	Czech	N,PR,SING,LOC	Naspk--

It is totally natural that the POS tag of a certain word is changed depending on the context. Therefore, we try to collect as much lexical information as possible for each lexical item from the corpus and store it as meta-information in the XML document. To store this information, we define two attributes for each word: (a) the '*gc*' (global context) attribute which contains various POS tags of a word in different contexts; (b) the '*lc*' (local context) attribute which represents the POS tag of the word in the local context.

We also lemmatize word forms by removing inflectional suffixes of nouns and adjectives automatically, such as plural, comparative, and superlative suffixes, and Ezāfe and indefinite clitics (if they have a written form). Since Persian has borrowed a large number of Arabic words, irregularities of noun plural forms and adjective superlative and comparative forms are unavoidable. These

[5]`http://nl.ijs.si/ME/`

$$\left[\begin{array}{ll}
\textsc{phon} & \langle \text{چک} \rangle \\
\textsc{synsem} & \left[\begin{array}{ll}
\textsc{loc} & \left[\begin{array}{ll}
\textsc{category} & \left[\begin{array}{ll}
\textsc{head} & \left[\begin{array}{ll} \textsc{type} & \textit{com} \\ \textsc{Abbr} & - \\ \textit{noun} & \end{array}\right] \\
\text{SUBJ} & \langle\rangle \\
\text{COMP} & \langle\rangle \\
\text{SPR} & \langle\rangle \\
\textit{category} &
\end{array}\right] \\
\textsc{content} & \left[\begin{array}{ll}
\textsc{index} & \left[\begin{array}{ll} \textsc{num} & \textit{sg} \\ \textsc{pers} & \textit{3rd} \\ \textit{index} & \end{array}\right] \\
\textsc{type} & \textit{empty} \\
\textit{content} &
\end{array}\right] \\
\textsc{context} & \\
\textsc{Ezāfe} & - \\
\textsc{Polarity} & + \\
\textit{loc} &
\end{array}\right] \\
\textit{synsem} &
\end{array}\right] \\
\textit{word} &
\end{array}\right]$$

Figure 5.5: Feature structure of the word 'چک'

cases are lemmatized semi-automatically. Verbs are lemmatized differently. The infinitive forms, and the past and present stems of each verb are defined semi-automatically for each verb.

To have a multi-functional data source, we define the types of the named-entities as well. Five named-entities, namely 'person', 'location', 'organization', 'time', and 'other', are defined in the data set. It should be added that in our data annotation, the named-entity 'time' is coarse-grained and it refers to any time expressions and dates, contrary to the BBN named-entity annotation[6] (Brunstein, 2002) in which time expressions and dates are more fine-grained, and they are recognized as separate named-entities to be used in a Question Answering system.

Examples 5.2 and 5.3 display the available meta-information for the noun 'چک' /ček/ 'Czech' and the verb 'دادن' /dādan/ 'give'.

(5.2) <w gc="Nasp---;Naspk--;Ncsp---;Ncsp--z" lc="Naspk--" clitic="empty" ne_sort="loc" lemma="چک">چک</w>

(5.3) <w gc="Nasp---;Ncsp---;Ncsp--z;Vpyssht----" lc="Vpyssht----" clitic="empty" inf_form="دادن" past_stem="داد" pres_stem="ده">داد</w>

In Chapter 2, we discussed the multi-token problem in Persian. Since our ultimate goal is to develop a treebank for Persian, the multi-token problem should be solved at both the syntactic and morphological levels. Clitics are the problematic cases at the syntactic level. If the clitic has a written form and it

[6]`http://catalog.ldc.upenn.edu/docs/LDC2005T33/BBN-Types-Subtypes.html`

is orthographically represented in the fused form, the clitic should split from its host, as in Examples 5.4 and 5.5. The orthographical hosts of the clitic 'ـش' /-aš/ in Example 5.4 is 'کتاب' /ketāb/ 'book' and it is 'جدید' /ǰadid/ 'new' in Example 5.5, whereas syntactically its host is the noun 'کتاب' /ketāb/ 'book' or the noun phrase 'کتاب جدید' /ketābe ǰadid/ 'new book'.

(5.4) کتابش
ketāb=aš
book=3SG
'his/her book'

(5.5) کتاب جدیدش
ketāb-e ǰadid=aš
book-EZ new=3SG
'his/her new book'

While splitting clitics from their hosts in the Bijankhan Corpus (Bijankhan, 2004), we found varieties of orthographical forms for copulative verbs and possessive and object pronouns as clitics. This variety is summarized in Table 5.2.

Table 5.2: Various orthographical forms of copulative verbs and possessive/object pronouns as clitics in the Bijankhan Corpus

	Copulative Verb	**Possessive/Object Pronoun**
1SG	ام، م، ستم	م، یم، ام، هام، شم، وم
2SG	ای، یی، ئی، ی، ستی	ت، یت، ات، هات، شت
3SG	ه، یه، ئه، ا، س، ست، یست	ش، یش، اش، هاش، شش
1PL	ایم، ییم، ئیم، ستیم	مان، یمان، امان، امون، شمان
2PL	اید، یید، ید، این، ین، ستید	تان، یتان، اتان، یتون، تون
3PL	اند، یند، ند، ان، ین، ن، ستند	شان، یشان، اشان، شون

To solve the multi-token problem at the morphological level, we use the method proposed by Müller (2010) to handle the Persian light verb construction. To exploit this idea and to be consistent in the entire corpus, we split the pre-verbal elements from the light verbs. The pre-verbal elements are mostly particles, such as 'باز' /bāz/ 'again' in Examples 5.6 and 5.7, which are POS tagged as prepositions in the Bijankhan Corpus.

(5.6) بازمی‌گشت.
bāz-mi-gašt
again-IMPF-turn.3SG
'He/she was returning.'

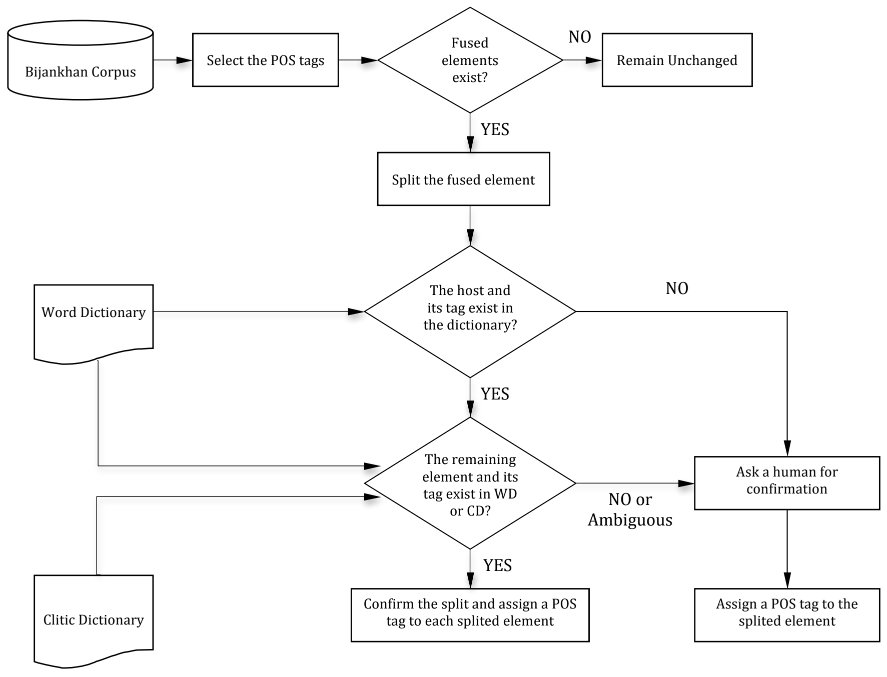

Bijankhan Corpus
Select the POS tags
Fused elements exist?
NO
Remain Unchanged
YES
Split the fused element
Word Dictionary
The host and its tag exist in the dictionary?
NO
YES
The remaining element and its tag exist in WD or CD?
NO or Ambiguous
Ask a human for confirmation
YES
Clitic Dictionary
Confirm the split and assign a POS tag to each splited element
Assign a POS tag to the splited element

Figure 5.6: Architecture of the algorithm used for splitting clitics

(5.7) بازگشته‌است.
bāz-gašte-ast
again-turned-3SG
'He/she has returned.'

We split clitics from their hosts semi-automatically. The architecture of the algorithm used for splitting clitics is represented in Figure 5.6. In this algorithm, first the POS tag of each word is checked to determine whether any fused elements exist in the word or not. If not, the word and its corresponding POS tag remain unchanged. If yes, the fused elements should split. To avoid any mistake in splitting these items, we use a hypothesis testing approach in our algorithm such that the fused elements are split into a host and the remaining element(s) with respect to the information in the POS tags. Next, this hypothesis is validated based on whether the host and the remaining element exist in the dictionaries (the Word Dictionary (WD) which contains the lexical items and the Clitic Dictionary (CD) which contains the clitics). If the answer to the hypothesis is 'yes', then the word which contains the fused element splits into sub-elements, and each sub-element is assigned an individual POS tag. Otherwise, the hypothesis is returned to a human and the algorithm asks the human to provide the correct split and assign the elements a correct POS tag. Human intervention is required for ambiguous and unknown cases.

When using the above algorithm to resolve the morphological multi-token problem in the Bijankhan Corpus (Bijankhan, 2004) described in 2.5.2, cases are still found that should be resolved either manually or automatically. A special POS tag is used in the Bijankhan Corpus for tokens like 'مادر' /mādar/ 'we in' in which two tokens are recognized as one unit. The above algorithm can be used for finding such units, splitting them into two tokens, and assigning each token an individual POS tag. A solution for handling multi unit tokens for treebanking is proposed in the following section.

Thus far, we have provided as much lexical information as possible for each lexical item, since in HPSG a huge amount of lexical knowledge is required. This information is useful for the next steps of the data annotation process.

5.4.2 Initialization Step

To initialize the annotation process, we require seed grammar rules as described in the bootstrapping process in Algorithm 1 on page 125. The seed grammar rules are a set of very basic grammar rules with high coverage defined and ordered hierarchically in the CLaRK system. To have binary branching in the trees and to define the dependency relations between the elements, either adjunct or complement, we extract bigrams from the Bijankhan Corpus. To this end, bigrams of the POS tags only, and bigrams of the words with their corresponding POS tags are extracted from the whole corpus, and the most frequent sequences are defined as seed regular grammars in the system.

Two sets of regular expressions are defined based on the bigrams. One set handles multi tokens in the pre-processing step, such as compound conjunctions, and compound verbs. The other set recognizes the most frequent constituents to initialize the data annotation process. Table 5.3 on the following page displays sample bigrams for multi tokens, and Table 5.4 shows sample bigrams for frequent constituents with their corresponding absolute frequencies from the Bijankhan Corpus written as regular expressions in CLaRK.

As shown in the tables, the morpho-syntactic and semantic information in the POS tags of the words is used for defining the grammar rules as regular expressions. It is possible to write productive grammar rules with a sequence of only POS tags, relatively productive grammar rules with partially lexicalized sequences, and restricted 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.

After defining a set of regular expressions as seed grammar rules, they are applied on the set of 50 sentences as seed data and their annotation are completed manually.

Table 5.3: Bigram samples for multi tokens written as regular expressions (RE)

Pattern	Absolute Freq.	Sample	Absolute Freq.	Translation	RE in CLaRK
Ncsp--- Vpyssht----	39213	مهاجرت کرد	10	moved	**Left RE** **RE:** <“Ncsp---”>,<“V%ys%%t----”> **Right RE** **RM:** <MV r=“1-001”> \ w< /MV>
Ncsp--- Vpyssh-f---	12011	انجام شده‌است	138	was done	**Left RE** **RE:** <“Ncsp---”>,<“V%ys%%-f---”> **Right RE** **RM:** <MV r=“1-002”> \ w< /MV>

Table 5.4: Bigram samples for constituents written as regular expressions (RE)

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** **RM:** <NPC r=“1-001”> \ w< /NPC>
Ncspt-z Urns---	6976	سال ۱۹۶۳	11	the year 1963	**Left RE** **RE:** <“Ncspt-z”>,<“Urns---”> **Right RE** **RM:** <NPC r=“1-002”> \ w< /NPC>

5.4.3 Main-processing Step

After annotating the set of 50 sentences as seed data, the main-processing and the post-processing steps are launched and continued iteratively on the remaining data (950 sentences), i.e. the main-processing and the post-processing steps follow each other and the both steps are processed in each iteration. The main processing step is done fully automatically such that in each iteration a set of k new rules (k=5) are extracted from the whole annotated data and added to the seed grammar rules as regular expressions in CLaRK. The updated seed grammar rules are then applied on the remaining unannotated sentences in the next iterations. Detailed description of this step together with the algorithm were already described in Section 5.3.

5.4.4 Post-processing Step

Similar to the main-processing step, the post-processing step is also performed iteratively such that in each iteration after the automatic annotation of sentences, a human annotator finishes the shallowly annotated sentences manually based on the annotation scheme described in Section 4.2. The output of this step is a set of fully annotated sentences, such as Figure 5.1 on page 121. The completely annotated sentences are used for extracting new grammar rules.

Similar to Marcus et al. (1993) and Simov et al. (2002a), only one analysis is provided for sentences with syntactic ambiguities in our treebank. The provided analyses rely on the most appropriate analyses based on the contexts.

5.5 Evaluation

In the initialization of the annotation process, 50 seed sentences are completed manually after being shallowly processed. Constraints are defined on the grammar rules to avoid their over-generation. The process of automatic and manual annotations of sentences in the main- and post-processing steps are continued iteratively. While more data is annotated in the bootstrapping process, in each iteration the rate of the grammar rules applied automatically and manually to complete the annotation of each sentence is recorded. Figure 5.7 displays the human annotation rate for each five iterations (for each 50 sentences). As can be seen in the figure, as the number of the grammar rules defined in the system grows, human effort to annotate the data decreases steadily which results in reducing human intervention in developing the data source. In the first iterations of the bootstrapping process, 74.05% of the analyses are done manually, and they are reduced to 39.01% in the last iterations. Since the length of sentences vary and longer sentences are more complicated than the shorter ones, the presented result is normalized with respect to the length of the sentences.

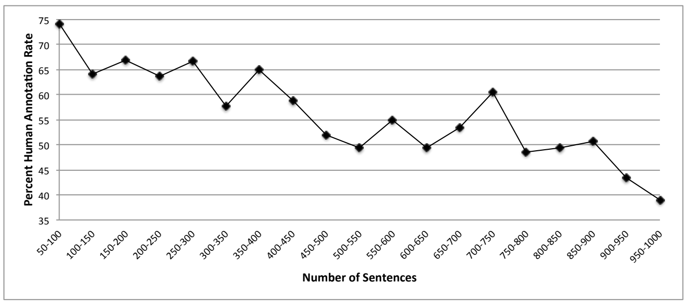


Figure 5.7: Human annotation rate in manual annotation

Even though the reduction of human annotation effort is significant, it is increased in some of the iterations due to the complexities of the sentences in these iterations. Consequently, their annotations require more human effort. As can be seen in the figure, there is a gradual decrease in manual annotation again in the next iterations which shows that the defined grammar rules of the seen contexts are enough for analyzing the sentences of these iterations.

In Table 5.5, the coverage of the rules (automatic vs. manual) is reported. To annotate this small treebank for Persian, which on average is 27 words long, on average 15.84 grammar rules (around 57.25% of the task) are applied manually for each sentence and the rest automatically.

Table 5.5: Summary of the bootstrapping result for the treebanking

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, precision, recall, and F-measure are computed. To compute precision in the equation (5.1), the correct grammar rules (GR) which are applied automatically against the total number of grammar rules applied automatically are measured:

$$Precision = \frac{number\ of\ correct\ automatic\ GR}{total\ \ number\ of\ automatic\ GR} \tag{5.1}$$

To compute recall in the equation (5.2), the correct grammar rules which are applied automatically are measured against the sum of correct automatic grammar rules and the manual grammar rules:

$$Recall = \frac{number\ of\ correct\ automatic\ GR}{number\ of\ correct\ automatic\ GR + manual\ GR} \tag{5.2}$$

F-measure in the equation (5.3) weights the interaction of precision and recall:

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{5.3}$$

The summary of the result is reported in Table 5.6. As can be seen, 86.20% of the applied automatic grammar rules are correct, and they cover 45.65% of all the automatic and manual grammar rules required to annotate the sentences.

Table 5.6: Evaluation results

F-measure	**Precision**	**Recall**
59.69	86.20	45.65

Based on the results, precision is relatively high, but recall is low for two reasons: (a) there are contexts that have not been seen so far, and the relevant grammar rules are not extracted as a result; (b) even if a grammar rule is extracted, there is no grantee that it is defined in the system during the bootstrapping process because of its low frequency. Of course annotating more sentences helps to increase recall, since having more annotated data and defining more grammar rules increase the coverage of the total grammar rules.

To verify this statement, we measure recall of our method for every 5 iterations and display it in Figure 5.8. As shown 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

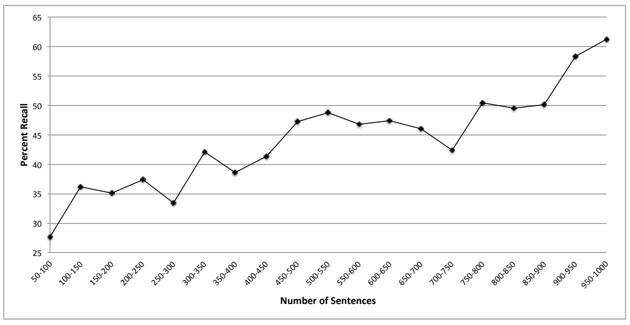


Figure 5.8: Recall measured every 5 iterations

in further iterations. There are two considerable drops in the graph. After consulting the data, we find that extraposition and ellipses frequently happened in this portion of data, and a higher amount of human intervention is required to complete the analyses. Moreover, errors in the assigned POS tags have a negative effect on the usage of the grammar rules. Additionally, changing the genre and the domain of the texts from politics to economy or sport news is another factor that affects recall.

The developed treebank is the preliminary data set which might require further editing because of changing the annotation scheme. Further editing of the developed treebank and re-segmenting sentences, in addition to deletion of the ungrammatical sentences, increased the number of trees to 1028 with a total number of 27026 word tokens. It is necessary to add that in 18.96% of the sentences extraposition is occurred.

5.6 Summary

In this chapter, we mainly described the development of an HPSG-based treebank for Persian. To this end, we used a bootstrapping approach for the data annotation. Developing this annotated data was the first attempt to build a treebank for Persian based on the HPSG formalism as its backbone.

In the first step, a set of seed grammar rules were defined as regular expressions in the CLaRK system and they were used for processing the 50 seed sentences shallowly. This initialization step was finished by completing the annotation of seed sentences manually. In the next steps of the annotation process (the main-processing and post-processing steps), a set of new sentences were automatically processed with this set of grammar rules and a human annotator completed the annotation of sentences manually in each iteration. To increase the automatic annotation, we extracted the grammar rules applied manually and iteratively augmented the seed grammar rules with the grammar rules applied frequently in the manual annotation.

The results showed that the proposed method could steadily reduce human intervention. It could be concluded that more seen contexts will result in more grammar rule definitions and gradual reduction of human intervention for further data annotation.

Chapter 6

Statistical Parsing of Persian

6.1 Introduction

One usage of treebanks is to train statistical parsers to build a grammar model and provide syntactic analyses for the new unseen data. In this chapter, we mainly describe the basic NLP tools and the data used for this goal. The tools are adapted to the Persian language when required. To represent the advantage of our developed treebank, this data set is converted into its parallel dependency-based treebank. The converted data is then used for training dependency parsers as well. The impact of the treebank annotation granularity on parsing performance is also taken into consideration when training a parser.

This chapter contains seven sections. Sections 6.2 and 6.3 briefly explain the basic properties of the NLP tools used for the rest of our study. These tools are the TnT and Stanford POS taggers, the Berkeley and Stanford constituency parsers, and the Malt and Mate dependency parses. Section 6.4 describes the data preparation steps for the constituency and dependency parsers. The data can potentially have three dimensions of annotation granularity, namely the lexical item, POS tag, and constituent label, which should be considered when training the tools. These three dimensions are defined in Section 6.5. In Section 6.6, we evaluate the performance of the parsers. Then, we study the effect of the annotation granularity on parsing. To this end, we change the data format from fine-grained to coarse-grained and compare the results. Since the parsers are trained with a small amount of data, they suffer from the data sparsity problem when used in a real application. The two levels for the data sparsity problem are the lexical and syntactic construction levels which are studied in more detail in Chapters 7 and 8. This chapter is summarized in Section 6.7.

The content of this chapter is mainly based on the following papers:

- Ghayoomi, Masood and Omid Moradiannasab (2012) "The effect of treebank annotation granularity on parsing: A comparative study" In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, November 30–December 1, 2012, Lisbon, Portugal, pp: 109–114.
- Ghayoomi, Masood (2013) "Introducing a treebank and a statistical parser for Persian" In *Proceedings of the 8th Conference of the Iranian Linguistics*, February 13–14, 2013, Tehran, Iran, pp: 666–679.
- Ghayoomi, Masood and Jonas Kuhn (2014) "Converting an HPSG-based treebank into its parallel dependency-based treebank" In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, May 26–31, 2014, Reykjavik, Iceland.

6.2 POS Tagging

POS tagging is the process of assigning a syntactic category to a word in its local context. There are two challenges that statistical POS taggers should usually deal with: (a) assigning a candidate POS tag to unknown words which are unseen in the training data, and (b) disambiguating the correct POS tag of a word if the word has more than one tag. Statistical methods are always sensitive to the data. As a result, the amount of the training data, and the difference between the training data and the test data are the variables that play important roles in this task. In this study, we use two POS taggers, namely the TnT and the Stanford POS taggers, that are described in this section.

6.2.1 TnT POS Tagger

The TnT POS tagger is developed by Brants (2000). This tagger is the implementation of the Viterbi algorithm (Viterbi, 1967) for second order Markov models. Moreover, a context-independent variant of the linear interpolation smoothing method (Manning and Schütze, 1999, p. 218) is used for dealing with the data sparsity problem. To handle the problem for unknown words in inflected languages, the suffix analysis method proposed by Samuelsson (1994) is used. In this method, the m last letters of a word are considered as a suffix and they are removed from the word forms to determine the base form of the word. To find the length of the suffix m, a maximum likelihood (Edwards, 1992) is estimated according to the frequencies of affixes derived from the corpus. To further reduce the data sparsity problem and improve the performance of the tagger, capitalization of the words is also taken into consideration in building the statistical model. The NLTK Toolkit[1] (Bird and Loper, 2004; Bird et al., 2009) contains the Python implementations of the TnT POS tagger.

6.2.2 Stanford POS Tagger

The Stanford POS Tagger[2], developed at the Stanford Natural Language Processing Group (SNLPG), is the Java implementation of a Maximum Entropy (MaxEnt) tagger which learns the log linear conditional probability model from a POS tagged text. A number of features are added to the feature set to handle the data sparsity problem for unknown words (Toutanova and Manning, 2000).

There exists a bi-directional dependency network tagger in the model which takes the local context into account to assign a POS tag. In this tool, the required information is extracted automatically from the training data as features, and it is used for tagging. Among the available options to extract the features, empirically we found that bidirectional features have the highest impact on the accuracy of the tagger for Persian words: the previous and the next word forms, two previous and two next word forms, the previous and the next POS tags, the current word form along with the POS tag of the previous and the next words, and the previous and the next bigrams.

To handle the data sparsity problem for unknown words, maximally 6 characters are determined as a prefix or suffix to realize the base form of a word. This property of the tagger is similar to the method used in the TnT tagger (Brants, 2000) except that the length of the affix is a predefined number, such as 6 characters, rather than a parameter to be set automatically.

6.3 Parsing

As mentioned, statistical parsers require annotated data to build the grammar model. The method used for creating this model has a direct effect on the performance of the parser. To this end, we describe two tools in this section.

6.3.1 Berkeley Parser

The Berkeley parser[3] is the Java implementation of an unlexicalized, constituency parser developed in the Berkeley Natural Language Processing Group (Petrov et al., 2006). This parser is a PCFG parser with latent annotation in the training data to refine grammar induction automatically such that the two-step method

[1]`http://nltk.org/`
[2]`http://nlp.stanford.edu/software/tagger.shtml`
[3]`http://nlp.cs.berkeley.edu/Software.shtml`

of 'splitting' and 'merging' is used for non-terminal symbols to maximize the likelihood (Edwards, 1992) of the training data. The Viterbi algorithm (Viterbi, 1967) is implemented inside the parser to find the best parse candidates. White-space is used for tokenization.

6.3.2 Stanford Parser

The Stanford parser[4] is the Java implementation of a lexicalized, probabilistic natural language parser developed by SNLPG (Klein and Manning, 2003). The parser is composed of a joint optimized PCFG and lexicalized dependency parser, and a lexicalized PCFG parser. There is a module in the parser that makes the parser able to learn the lexical items and to assign them POS tags while parsing a raw text. The most important function of this module is to calculate the probability of a word given its tag, $P(word|tag)$, to let the parser choose the best tag for the word in the local context, and utilize this probability to find the best tree structure for a sentence. It should be added that the Max-Ent POS tagger, described in Section 6.2.2, is implemented within the Stanford CoreNLP package as well for assigning POS tags. In this parser, white-space is used for tokenization. There is a built-in Viterbi algorithm (Viterbi, 1967) in the parser to select the k best parse results (Huang and Chiang, 2005). Following the study of Collins (1999), heads have to be provided for the parser and they play a major role in the chart parse.

The evaluation of the parsing result is done with Evalb[5] to report the standard bracketing metric results like precision, recall, and F-measure as well as exact matching.

In the adaptation of the Stanford parser for Persian, we provide a head table for the parser developed semi-automatically based on the head-daughter relations defined in the treebank. To resolve the multi token problem and the tokenization, we implement the pre-processing step described in Section 5.4.1 within the parser.

6.3.3 Malt Parser

The Malt parser[6] is a data driven dependency parser that uses a transition-based approach but not probabilities for dependency parsing (Nivre et al., 2006). The inductive dependency parsing introduced by Nivre (2005b) is implemented in the parsing algorithm. A history-based feature model is also used for predicting the next parser action. The embedded classifiers of the learning algorithm,

[4]`http://nlp.stanford.edu/software/lex-parser.shtml`
[5]`http://nlp.cs.nyu.edu/evalb/`
[6]`http://www.maltparser.org/`

namely the Support Vector Machine (SVM) and large linear classification, use the features that are extracted from the training data. The input data format of this parser is the CoNLL 2006 shared task[7].

6.3.4 Mate Parser

The Mate parser[8] is another data driven dependency parser that uses the MST approach for dependency parsing (Bohnet, 2009). In the parser, the second order MST parsing algorithm introduced by Eisner (1996) is implemented. The exploited parsing algorithm is a bottom-up parsing algorithm which is similar to the CKY chart parsing algorithm. In this parser, instead of computing probabilities, syntactic and semantic features are extracted as vectors to be used by the SVM classifier. The dependency score of a sentence is the sum of the scores on all of the edges in that sentence, therefore no probability is used. The score of the edge is the product of the feature vector representation for each edge with a weight vector (Bohnet, 2009). This parser is able to do semantic parsing as well by assigning semantic role labels. The input data format of this parser is the CoNLL 2009 shared task[9].

6.4 Data Preparation for Parsing

In Section 6.3, two types of statistical parsers were introduced: a constituency parser and a dependency parser. In this section, we describe the data preparation steps for both types of the parsers.

6.4.1 Constituency-based Data

To prepare the required data format for the Berkeley and Stanford parsers, the Persian treebank has to be normalized and converted from the XML format into the plain text Penn Treebank style. To this end, several conversions are done on the treebank. As said, Persian is a right-to-left language, and since neither the Berkeley nor the Stanford parsers supports bidirectional parsing, we have to convert the treebank into the left-to-right direction, similar to what is done in the Penn Arabic Treebank experiment[10] without losing any information as displayed in Figure 4.3 on page 78.

[7]`http://ilk.uvt.nl/conll/#task`
[8]`http://code.google.com/p/mate-tools/`
[9]`http://ufal.mff.cuni.cz/conll2009-st/task-description.html`
[10]`http://www.ircs.upenn.edu/arabic/`

Figures 6.1 and 6.2 on the next page display the right-to-left and left-to-right tree representations of Example 6.1.

(6.1) بورن به‌دنبال این است که چیزی را بسازد که قبلاً وجود نداشته‌است.

born	bedonbāl-e	in	ast	ke	čiz-i	rā
Born	after-EZ	this	is	that	something-RES	DOM

be-sāz-ad	ke	qablan	vojud	na-dāšte-ast
SUBJ-create-3SG	that	before	existence	NEG-had-3SG

'Born is after this [namely] to create something that did not exist before.'

In this conversion, the constituents' hierarchies have to remain unchanged. Additionally, since we want to train the parsers with traceless trees, the '*nid*' nodes must be removed from the trees. Before removing the nodes, the mother node should rename as $X-$nid, where X is the label of the mother node. The label '$-$nid', which functions as a slashed element in HPSG, is propagated to the parent nodes. The extraposed elements which are considered as slashed elements are bound at the topmost node by the Head-Filler-Schema as represented in Figure 6.3.

After converting the XML format of the trees into the plain text Penn Treebank style displayed in Figure 6.4 on page 146, the data is used for training the constituency parsers. Moreover, in the normalization process, the following information is lost from the original treebank: structure sharing, the links of the extraposed elements to their associated canonical positions, the *Pragmatic* node, the attributes and the values of named entities, lemmas, and the attributes for determining types of 'که' /ke/ 'that' and 'ی/' /-i/ and clitics. Since it is possible for white-spaces to be used between the elements of a word, the white-spaces are replaced with '$-$s$-$' to recognize multi tokens as one unit and to solve the tokenization problem.

In the treebank, a hierarchical analysis is provided for coordination phrases. To be more precise in the analyses of this syntactic construction, we add a functional label to the symmetric coordination phrase to determine the type of the coordination phrase according to its dependents.

6.4.2 Dependency-based Data

To train the dependency parsers, we need a dependency treebank. Despite the existing Persian dependency treebanks, namely UPDT and PDT developed by Seraji et al. (2012a) and Rasooli et al. (2013) respectively, there are reasons that motivated us to use our treebank to train a dependency parser.

One major advantage of our treebank is the ease with which it can be converted to another grammar formalism, such as a dependency grammar. We aim

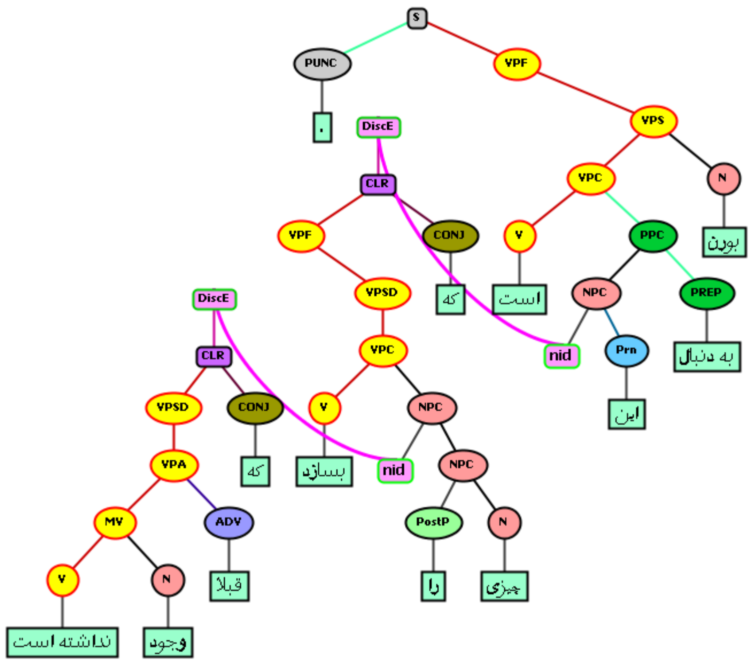


Figure 6.1: Right-to-left tree representation of Example 6.1

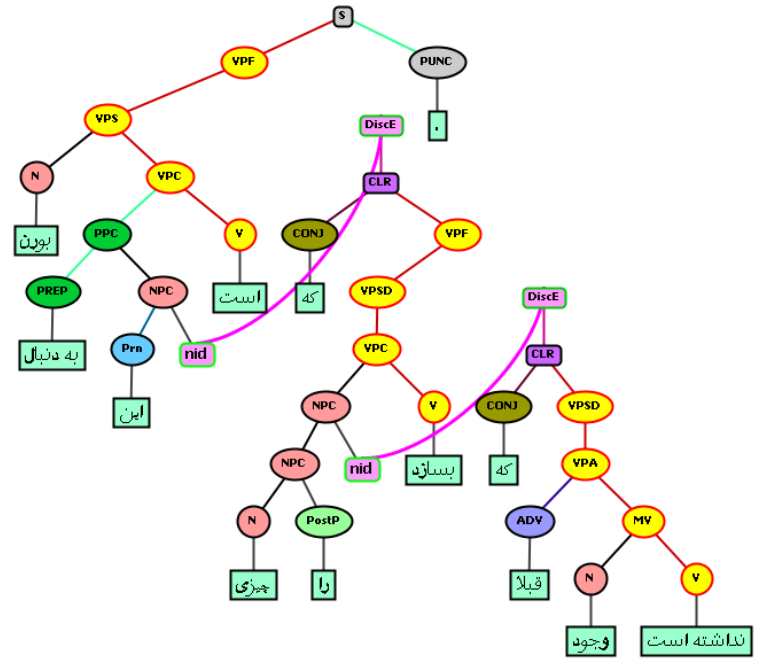


Figure 6.2: Left-to-right tree representation of Example 6.1

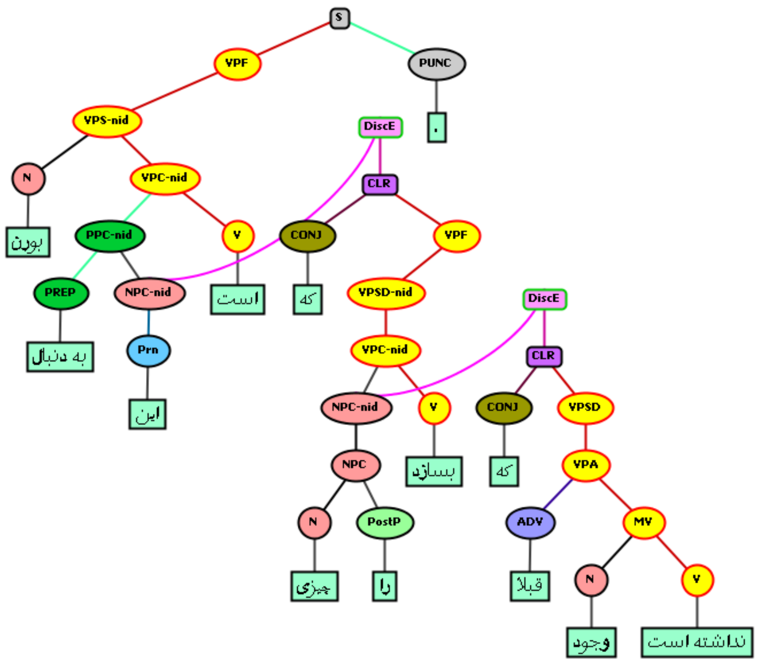


Figure 6.3: Traceless left-to-right tree representation of Example 6.1

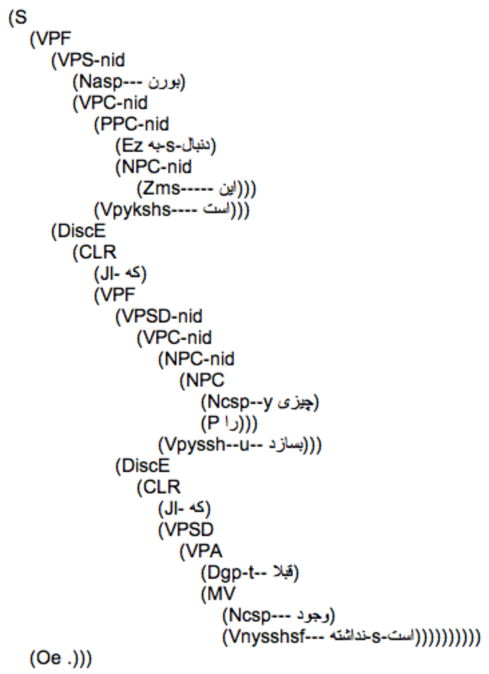

```
(S
  (VPF
    (VPS-nid
      (Nasp--- بورن)
      (VPC-nid
        (PPC-nid
          (Ez به-s-دنبال)
          (NPC-nid
            (Zms----- این)))
        (Vpykshs---- است)))
    (DiscE
      (CLR
        (Jl- که)
        (VPF
          (VPSD-nid
            (VPC-nid
              (NPC-nid
                (NPC
                  (Ncsp--y چیزی)
                  (P را)))
              (Vpyssh--u-- بسازد)))
          (DiscE
            (CLR
              (Jl- که)
              (VPSD
                (VPA
                  (Dgp-t-- قبلا)
                  (MV
                    (Ncsp--- وجود)
                    (Vnysshsf--- نداشته-s-است)))))))))
  (Oe .)))
```

Figure 6.4: The Penn Treebank style for tree representation of Example 6.1

Algorithm 2 Converting a constituency treebank into a dependency treebank

```
Input: A tree of a sentence from the HPSG-based treebank for Persian,
       The Head Table (HT),
       The modified version of Dependency Relations (DR) based on the Stanford
       Typed Dependencies (de Marneffe and Manning, 2008)
if Projective option selected then
  repeat
    Step1: Traverse the tree to find a pair of sibling leaves
    Step2: Select the head node of the sibling leaves based on HT
    Step3: Detect the DR between the sibling leaves based on the head-daughter
           dependency relations
    Step4: Write the corresponding DR according to the CoNLL format
    Step5: Remove both of the leaves and transfer the head information to the
           parent node
  until The tree root node is met
else if Non-projective option selected then
  repeat
    Do Step1
    if The parent node has a slashed element then
      Transfer the information of the leaf node along with the slashed element to
      the parent node as ‘np-head’
    else if The parent node has the Head-Filler relation then
      Use the ‘np-head’ of the leaf node with a slashed element instead of the head
    end if
    Do Steps 2 to 5
  until The tree root node is met
end if
Write the ROOT relation for the head of the tree
Sort the CoNLL format according to the sequence of the words in the input sentence
```

at showing this advantage practically. To this end, we reformat the XML data format into the CoNLL shared task format in such a way that each lexical entry of the sentence is on one line and its relevant morpho-syntactic and semantic information is organized in separate columns (10 columns in the CoNLL 2006 data format, and 14 columns in the CoNLL 2009 data format). Algorithm 2 is employed to convert our constituency treebank into its parallel dependency treebank.

In the conversion process, we employ a depth-first searching approach to identify the constituents. In this process, each tree is traversed to find sibling leaves. A dependency relation is defined for each pair of sibling leaves in which the head of the pair is determined in a head table. The head table is provided by extracting the grammar rules from the original treebank. After defining each pair of sibling leaves as a dependency relation, the leaves are removed, and the information of the head node is transferred to the parent node. The process of removing the leaves in each step produces new leaves which are in fact the old parents, and they must be processed in the next steps. The conversion steps continue in a loop to process all nodes of the tree, and the algorithm halts when the root of the tree is reached.

Since the positions of the extraposed elements as slashed elements are determined in the trees (displayed by the '−nid' functional label on trees) and they are bound at the topmost node by the Head-Filler-Schema, it is possible to convert the projective trees into non-projective trees. After traversing a tree to find a pair of sibling leaves, if a parent node contains a slashed element, the information of the slashed element in addition to the head node is transferred to the parent node. To draw a non-projective tree, it is important to keep the information of the leaf node that has a slashed element. This information, which is related to the head of a non-projective tree, is temporally kept in a variable, which we called it 'np-head', and it is transferred to the upper nodes until the Head-Filler relation is met. By binding the slashed element, the 'np-head' information is retrieved, and it is used for defining the dependency relation.

In the dependency conversion, besides defining the dependents, the types of dependencies should be defined. Contrary to the constituency-based treebanks, which consist of intermediate nodes that contain the type of the dependency relations, in the converted dependency-based treebank no intermediate node exists. Consequently, we modify the Stanford Typed Dependencies (de Marneffe and Manning, 2008) and exploit the modified version to define the types of the dependency relation in our dependency-based treebank instead of the basic dependency relations in the original treebank. Appendix D demonstrates the hierarchy of the defined dependency relations in the dependency conversion.

In the conversion, we rely only on the linear word order without inserting any empty categories like ellipses or traces. Figures 6.5 and 6.6 on the next page display the projective and non-projective dependency trees of Example 6.1 on page 144 respectively. For convenience, this example is repeated in Example 6.2. The non-projective equivalent data formats based on the CoNLL 2006 and 2009 shared tasks are represented in Figures 6.7 and 6.8.[11]

[11]Each column of the CoNLL 2006 and 2009 data format contains certain information introduced as follows: ID contains the index of the word in the linear order; FORM contains the word form; LEMMA contains the lemma of the word; CPOSTAG contains the coarse-grained POS tag; FPOSTAG contains the fine-grained POS tag; FEATURE contains morphosyntactic features; DEPREL contains the dependency relation; HEAD contains the index of the word that functions as a head; PHEAD contains the projective head; PDEPREL contains the dependency relation to the projective head; PL contains the predicted lemma; PT contains the POS tag; PPT contains the predicted POS tag; F contains features; PF contains the predicted feature; PH contains the predicted head; PDR contains the predicted dependency relation; SR contains the semantic role; PSR contains the predicted semantic role.

(6.2) بورن به‌دنبال این است که چیزی را بسازد که قبلاً وجود نداشته‌است.

born bedonbāl-e in ast ke čiz-i rā
Born after-EZ this is that something-RES DOM
be-sāz-ad ke qablan vojud na-dāšte-ast
SUBJ-create-3SG that before existence NEG-had-3SG
‘Born is after this [namely] to create something that did not exist before.’

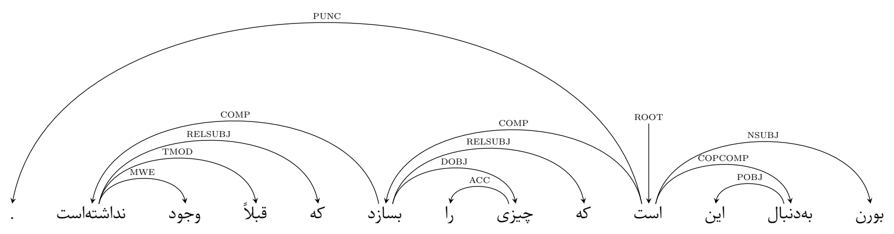

PUNC
COMP
RELSUBJ
TMOD
MWE
COMP
RELSUBJ
DOBJ
ACC
ROOT
NSUBJ
COPCOMP
POBJ
. نداشته‌است وجود قبلاً که بسازد را چیزی که است این به‌دنبال بورن

Figure 6.5: Projective dependency relation of Example 6.2

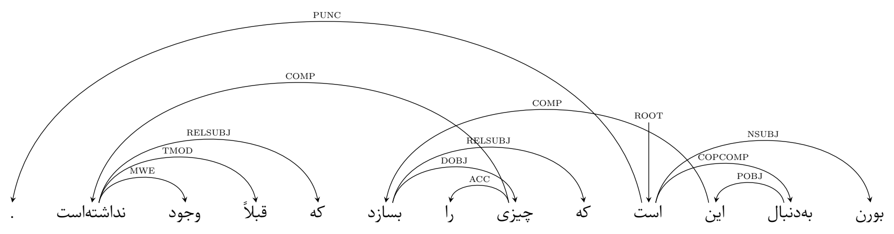

PUNC
COMP
COMP
ROOT
RELSUBJ
TMOD
MWE
RELSUBJ
DOBJ
ACC
NSUBJ
COPCOMP
POBJ
. نداشته‌است وجود قبلاً که بسازد را چیزی که است این به‌دنبال بورن

Figure 6.6: Non-projective dependency relations of Example 6.2

ID	FORM	LEMMA	CPOSTAG	FPOSTAG	FEATURE	HEAD	DEPREL	PHEAD	PDEPREL
1	بورن	بورن	N	Nasp---	asp---	4	NSUBJ	_	_
2	به‌دنبال	به‌دنبال	E	Ez	z	4	COPCOMP	_	_
3	این	این	Z	Zms-----	ms-----	2	POBJ	_	_
4	است	بودن	V	Vpykshs----	pykshs----	0	ROOT	_	_
5	که	که	J	Jl-	l-	8	RELSUBJ	_	_
6	چیزی	چیز	N	Ncsp--y	csp--y	8	DOBJ	_	_
7	را	را	P	P	_	6	ACC	_	_
8	بسازد	ساختن	V	Vpyssh--u--	pyssh--u--	3	COMP	_	_
9	که	که	J	Jl-	l-	12	RELSUBJ	_	_
10	قبلاً	قبلاً	D	Dgp-t–	gp-t–	12	TMOD	_	_
11	وجود	وجود	N	Ncsp---	csp---	12	MWE	_	_
12	نداشته‌است	داشتن	V	Vnysshsf---	nysshsf---	6	COMP	_	_
13	.	.	O	Oe	e	4	PUNC	_	_

Figure 6.7: CoNLL 2006 non-projective dependency format of Example 6.2

ID	FORM	LEMMA	PL	PT	PPT	F	PF	HEAD	PH	DEPREL	PDR	SR	PSR
1	بورن	بورن	_	N	_	asp---	_	4	_	NSUBJ	_	_	_
2	به‌دنبال	به‌دنبال	_	E	_	z	_	4	_	COPCOMP	_	_	_
3	این	این	_	Z	_	ms-----	_	2	_	POBJ	_	_	_
4	است	بودن	_	V	_	pykshs----	_	0	_	ROOT	_	_	_
5	که	که	_	J	_	l-	_	8	_	RELSUBJ	_	_	_
6	چیزی	چیز	_	N	_	csp--y	_	8	_	DOBJ	_	_	_
7	را	را	_	P	_	_	_	6	_	ACC	_	_	_
8	بسازد	ساختن	_	V	_	pyssh--u--	_	3	_	COMP	_	_	_
9	که	که	_	J	_	l-	_	12	_	RELSUBJ	_	_	_
10	قبلا	قبلا	_	D	_	gp-t--	_	12	_	TMOD	_	_	_
11	وجود	وجود	_	N	_	csp---	_	12	_	MWE	_	_	_
12	نداشته‌است	داشتن	_	V	_	nysshsf---	_	6	_	COMP	_	_	_
13	.	.	_	O	_	e	_	4	_	PUNC	_	_	_

Figure 6.8: CoNLL 2009 non-projective dependency format of Example 6.2

In the followings, the general properties of the Persian dependency treebanks, such as the UPDT developed by Seraji et al. (2012a) and the PDT developed by Rasooli et al. (2013) are briefly described, and they are compared with our converted dependency-based treebank. In the comparison, we focus on the following properties: the size of the treebanks, the number of the defined dependency relations, the projectivity of trees, and the syntactic properties, such as POS tags, lemmatization, clitics, multi tokens, determiner phrases, and coordination phrases along with their corresponding annotation schemes.

Treebank Quantity and Dependency Relations: currently our treebank contains 1,028 sentences (27,026 word tokens), and 49 labels are used for defining the dependency relations in the sentences. We modified the Stanford Typed Dependencies (de Marneffe and Manning, 2008) and utilized the modified version to define the dependency relations in our treebank. The current online release of UPDT contains 6,000 sentences (151,671 word tokens). The dependency relations are defined by 102 labels in this treebank. The dependency relations defined in this treebank are also a modified version of the the Stanford Typed Dependencies (de Marneffe and Manning, 2008). The PDT contains 29,982 sentences (498,081 word tokens), and 46 labels are used for defining the dependency relations.

Projectivity vs Non-projectivity: our dependency treebank can have both projective and non-projective trees for discontinuous constructions, whereas only non-projective analyses exist in the UPDT and the PDT .

POS Tag and Lemmatization: in the three treebanks, morpho-syntactic information of the words is available, but they differ in the degree of granularity of the information. Our treebank contains more fine-grained POS tags than the other two treebanks. In terms of the availability of lemma information, our treebank and the PDT contain lemmas, but the words in the UPDT are not lemmatized.

Clitic: in our treebank, the morpho-syntactic information of Ezāfe and the post-nominal indefinite determiner are defined in the POS tags of hosts, but this information is overlooked in the UPDT and the PDT. In our treebank, we provide accurate syntactic analyses for possessive and object clitics as well as copulative verb clitics by splitting them from their hosts, as described in Section 5.4.1, whereas in the UPDT and the PDT no analyses are provided for the clitics, and in their analyses, the dependency relations are provided for the host and these clitics are not analyzed accurately.

Multi tokens: our treebank provides an internal syntactic analysis for the light verb construction, while light verb constructions are mostly considered as one token in the PDT except the ones that there is a long distance between the pre-verbal element and the light verb. In these cases, the pre-verbal element dominates the intervening elements as its argument.

Determiner Phrase: in the annotation scheme of our treebank, a determiner is the head of a determiner phrase, but this is not the case in the UPDT and the PDT where a noun is the head. Figure 6.9 on the next page illustrates the dependency tree of Example 6.3 based on our annotation scheme where a determiner is the head. Figures 6.10 and 6.11 illustrate the analyses of Example 6.3 according to the annotation scheme of Seraji et al. (2012a) and Rasooli et al. (2013).

Coordination Phrase: a coordinator is the head of a coordination phrase in the annotation scheme of our treebank, as represented in Figure 6.9, whereas in the UPDT, a coordinator is not a head at all, and in the PDT, a coordinator is the head of the sister element. Figures 6.10 and 6.11 illustrate the annotation scheme of a coordination phrase in the UPDT and the PDT, respectively.

(6.3) The boy ate the apple and the orange.

6.5 Annotation Granularity for Parsing

The state-of-the-art statistical parsers trained with treebanks (Collins, 1999; Charniak, 2000) are mainly developed based on PSG. The POS tags of the words in the treebanks are defined according to a tag set which contains the syntactic categories of the words with the optional morpho-syntactic information. Moreover, the constituent labels in treebanks might also be enriched with syntactic functions. The developed annotated data in the framework of a deeper formalism, such as HPSG, provides a fine-grained representation of the linguistic knowledge. The performance of the parsers trained with such fine-grained information has not beaten the state-of-the-art results (Miyao, 2006) due to the complexities that are added to the parsers. We believe that different dimensions and levels of the annotation granularities affect the parsing performance. To study this assumption, we consider three dimensions, namely lexical item, POS tag, and constituent label, and the two levels of fine- and coarse-grained annotations.

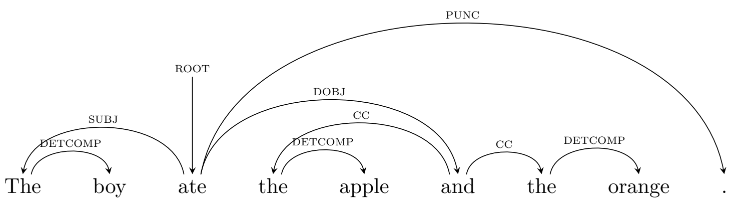


Figure 6.9: Dependency representation of Example 6.3 according to our annotation scheme

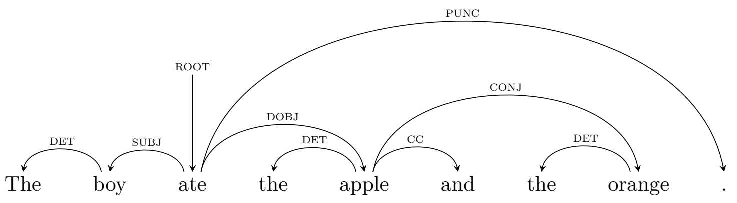


Figure 6.10: Dependency representation of Example 6.3 according to Seraji et al. (2012a) annotation scheme

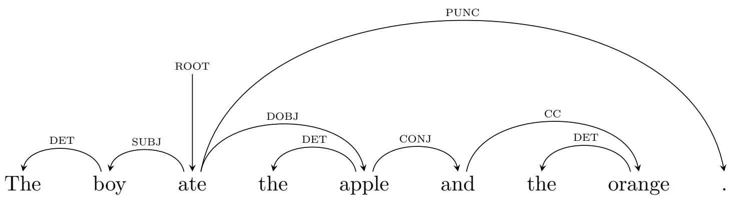


Figure 6.11: Dependency representation of Example 6.3 according to Rasooli et al. (2013) annotation scheme

6.5.1 Lexical Item

The words of a language represent fine-grained concepts, and they play a very important role in lexicalized, statistical parsers. Since the sparseness of data is the biggest challenge in data oriented parsing, statistical parsers will perform poorly if they are trained with a small set of data, or when the domain of the training and the test data is not similar. To represent coarse-grained concepts, a clustering approach is proposed in Chapter 7 to handle the data sparsity problem at the lexical level. In Section 7.6.2, we will study the impact of coarse-grained representation of lexical items on parsing.

6.5.2 POS Tag

The syntactic categories of words at the sentence level are the very basic linguistic knowledge that the parser learns. Therefore, they play a very important role for of a parser. The quality of the assigned POS tags to the words and the amount of information that they contain have a direct effect on the performance of the parser. The representation of this knowledge can be either coarse-grained, such as Noun, Verb, Adjective, etc., or fine-grained to contain morpho-syntactic and semantic information, such as Noun-Single, Verb-Past, Adjective-Superlative, etc. The fine-grained representation of the POS tags increases the tag set size and intensifies the complexity of the tagging task for a statistical POS tagger to disambiguate the correct labels. Moreover, it provides detailed information for the parser which can help the parser to provide more accurate analyses of sentences.

6.5.3 Constituent Label

Depending on the formalism used as the backbone of a treebank, the labels of the nodes at the phrasal level can be either fine- or coarse-grained. The annotated data in the Penn Treebank (Marcus et al., 1993) provides relatively coarse-grained constituent labels in which mostly the types of the phrasal constituents like NP, VP, etc. are determined. However, in the latest version of the Penn Treebank, the syntactic functions are added to the labels as well, but this information is not available for all nodes. In contrast, annotated data developed based on a deep formalism like HPSG, such as the BulTreeBank (Simov et al., 2004), provides fine-grained representations of constituent labels since the types of head-daughters' dependencies are defined explicitly for all nodes. Representation of dependency information on constituents adds complexities to a parser for disambiguating the type of the dependency relation as the size of the constituent label set increases. This information, however, can be very helpful for applications that require more fine-grained analysis of the syntactic relations.

6.6 Evaluation

6.6.1 Experimental Setup

Parsers can potentially be used and evaluated in three different scenarios: (a) using raw sentences, (b) using POS tagged sentences which are tagged with an external POS tagger, or (c) using sentences with the assigned gold POS tags. In the following, the empirical results of the constituency parsing scenarios, as well as the dependency parsers are reported and discussed. In Section 3.4.3, a number of parsing evaluation metrics were introduced. Among them, we use the labeled PARSEVAL metric, EM, and LA for evaluating the performance of the constituency parsers. The labeled- and unlabeled attachment metrics are used as the evaluation metrics for the dependency parsers. Additionally, the obtained results for studying the impact of annotation granularity on parsing are reported and discussed.

Since there is no further gold standard data to evaluate the parsing performance, we divide the sentences in our developed treebank into non-overlapped training and test sets, without filtering the sentences in terms of their length. To this end, 10-fold cross validation is used such that in each fold only 10% of the treebank is considered as test data and the remaining as training data.

Since the treebank is small, a parser will face the data sparsity problem for unknown words. Smoothing methods (Manning and Schütze, 1999, pp. 196–221) can be used to handle this problem. In all of our experiments, no smoothing technique is used so as to have reliable performance and to be able to compare the performance of the parsers.

One problem that the parsers has is that they do not handle ellipses. Even though these elements exist in training data of the constituency parsers, none of them are able to predict and insert ellipses. Consequently, the sentences that contain ellipsis (7% of the data) are excluded in the evaluation of the parser performance and the reported results are based on the remaining data (93% of the data).

6.6.2 Results and Discussion

Constituency Parsing: Scenario I

In Scenario I, raw sentences are used as input; i. e. no POS tags are assigned to the input data. The parser first must use a built-in POS tagger trained with the treebank data to assign POS tags to the words, and then it performs parsing. In a real application, usually, the parsing model in Scenario I is used. The performance of the Berkeley and Stanford parsers is reported in Table 6.1.

Table 6.1: Parsing accuracy using raw data

Tool	**F-measure**	**Precision**	**Recall**	**EM**	**LA**
Berkeley Parser	54.95	55.15	54.76	5.219	81.57
Stanford Parser	46.90	46.93	46.86	3.229	82.11

As can be observed from the results, in Scenario I the Berkeley parser outperforms the Stanford parser in F-measure, while the Stanford parser has a slightly higher performance based on the LA measure.

Table 6.2: Test data information

Granularity	**Num. of POS Labels**	**Num. of Word Types**	**Num. of Word Tokens**	**Num. of Unknown Words**
fine	248	5700	27026	513
coarse	15			

Table 6.3: Tagging accuracy of the TnT and Stanford POS taggers

Tool	**Granularity**	**Accuracy** (%)
TnT	fine	81.48
	coarse	95.70
Stanford	fine	95.90
	coarse	98.38

After checking the number of sentences parsed by the parsers, we realized that out of 1,028 sentences, the Stanford parser parsed all of the sentences, but the Berkeley parser parsed around 90% of the data (920 sentences), which demonstrates that the Stanford parser is more robust to be able to provide analyses for all sentences.

Constituency Parsing: Scenario II

Although Persian suffers from a lack of available treebanks for parsing, there is a large amount of POS tagged data that can be used for training POS taggers and creating more accurate tagging models to achieve a higher performance. In Scenario II, raw sentences are first POS tagged by an external POS tagger, then the output of the tagger is used as the input to the parser in a pipeline.

For tagging, we first train the TnT and Stanford POS taggers with the Bijankhan Corpus (Bijankhan, 2004) and test them with the sentences in our treebank. The shared data between the Bijankhan Corpus and the treebank is excluded from the training data so as to have no overlap between the test and training data. The information about the test data is summarized in Table 6.2. The accuracy of both the TnT and Stanford POS taggers are reported in Table 6.3.

To study the impact of the annotation granularity on POS tagging described in Section 6.5.2, the fine-grained POS tags of both training and test data are converted into coarse-grained POS tags and the tools are retrained and evaluated. As shown in the results, coarse-grained POS tagging obtains higher accuracy than the fine-grained POS tagging. This determines that the complexity of fine-grained annotation has a negative effect on the performance of the POS tagger. Comparing the results of the taggers, it can be observed that the Stan-

Table 6.4: Parsing accuracy tested on the output of the tagger

Tool	F-measure	Precision	Recall	EM	LA
Berkeley Parser	57.59	57.53	57.66	6.9	81.90
Stanford Parser	52.43	52.56	52.30	4.167	82.30

Table 6.5: Parsing accuracy with gold POS tags

Tool	F-measure	Precision	Recall	EM	LA
Berkeley Parser	62.27	62.25	62.28	8.485	83.51
Stanford Parser	59.42	59.44	59.40	5	85.65

ford tagger has outperformed the TnT tagger in both coarse- and fine-grained tagging.

Since in HPSG lexical knowledge plays a very important role, we use the Stanford tagger with fine-grained POS tags in our next experiments. In the parsing step, the constituency parsers are trained with the treebank to build the grammar model. Table 6.4 represents the performance of the parsers. In Scenario II, the Berkeley parser outperforms the Stanford parser in F-measure, while the Stanford parser has a slightly higher performance based on the LA measure. Checking the output of the parser, we realized that the Berkeley parser was able to parse around 91% of the test data (927 sentences), while the Stanford parser provided analyses for all sentences.

Constituency Parsing: Scenario III

When we checked the output of the parsers in Scenario II, we found that the low performance of the POS tagger has a negative impact on parsing due to the errors in the POS tags assigned to both known and unknown words. Since the assigned incorrect POS tags propagate in the parsing step, the accuracy of the parser reduces. We experiment with another scenario (Scenario III) in which the sentences with the assigned gold POS tags are given to the parsers and the parsers are forced to use these POS tags. The performance of the parsers based on Scenario III is reported in Table 6.5. As can be observed, the parsing model in Scenario III outperforms Scenarios I and II which determines the importance of the quality of the assigned POS tags and also the accuracy of the tagger. To avoid the negative impact of tagging on parsing, and to only focus on parsing in our study, we use the gold POS tags in training data for the rest of our experiments.

Although, in general, the performance of the Berkeley parser is better than the Stanford parser, the Berkeley parser has a poor performance using the LA

Table 6.6: Dependency results of the constituency parsers

Learning Scenario	**Tool**	**Labeled Attachment**	**Unlabeled Attachment**
Scenario I	Berkeley Parser	59.63	67.63
	Stanford Parser	56.75	65.74
Scenario II	Berkeley Parser	63.06	70.39
	Stanford Parser	57.58	66.55
Scenario III	Berkeley Parser	66.96	73.36
	Stanford Parser	63.69	72.44

metric in the three parsing scenarios. This difference indicates that the errors on choosing the constituent labels by the Berkeley parser is higher than the Stanford parser. This means that the output of the Berkeley parser compared with the gold data requires a higher degree of editing. Kübler et al. (2008) had a similar finding when they compared the performance of three constituent parsers trained with two German treebanks.

After checking the number of parsed sentences for each parser, we realized that out of 1,028 sentences, around 90% of the data (920 sentences) is parsed by the Berkeley parser which shows that due to the small amount of the training data, the parser is not robust enough to provide analyses for all sentences. In contrast, the Stanford parser is a robust parser which provided parse trees for all of the sentences. Although the F-measure of the Berkeley parser is higher than the Stanford parser, the robustness of the parser is very important for further steps of our task, especially the application of the parser for active learning described in Chapter 8. Therefore, we chose the Stanford parser as the parsing tool for the rest of our experiments.

Dependency Evaluation and Parsing

In Section 3.4.3, we discussed a dependency-based evaluation metric to evaluate the performance of a constituency parser by converting the constituency trees into dependency relations. To evaluate the performance of the constituency parsers in Scenarios I, II, and III described above, we convert the output of the parsers as well as the test data into their parallel dependency relations by using Algorithm 2 on page 147, and then we evaluate the performance of the parser based on the dependency relations. Since the constituent trees are projective, in the dependency evaluation we also use projective trees. Table 6.6 represents the dependency evaluation of the trees in Scenarios I, II, and III.

In the model proposed in Chapter 8, we require a dependency parser for a part of our study. Therefore, we need to train a dependency parser by converting the constituency treebank into its parallel dependency treebank using the

Table 6.7: Dependency parsing results by using gold POS tags

Tool	**Projectivity**	**Labeled Attachment**	**Unlabeled Attachment**
Malt	projective	77.07	82.63
	non-projective	76.63	82.04
Mate	projective	80.21	84.86
	non-projective	80.17	84.38

Algorithm 2 on page 147. The converted data, containing either projective or non-projective trees along with gold POS tags, is used for training and testing the Malt and Mate dependency parsers. Table 6.7 reports the performance of both dependency parsers as the upper bound with projective and non-projective trees.

As can be observed, the Mate parser outperforms the Malt parser in both projective and non-projective trees. Additionally, both of the parsers achieve higher performance when the attached dependencies are unlabeled. Comparing the results of the parsers in terms of projectivity, the performance of both parsers is to some extent decreased in non-projective parsing mode. It must be mentioned that 0.8% of the dependency relations in our dependency-based treebank are non-projective which have a slight negative impact on the overall performance of the parser compared with the parser trained with projective trees. We use the projective Mate parser for our experiments in Chapter 8.

Annotation Granularity Results

Section 6.5 described three possible annotation dimensions for parsing. In the following, we describe the data preparation and the experimental setup to study the effect of each annotation dimension on parsing performance.

To provide a coarse-grained representation of morpho-syntactic information of words, only the main POS tags of the words (15 POS tags) are used rather than all the information available in the POS tags. To provide simple head-daughter relations as coarse-grained constituent labels, only the type of the dependency relations from the constituency-based data are removed like the dependency labels for Head-Adjunct and Head-Complement relations as well as the type of clauses, without any changes on other head-daughter relations.

In the first step of our experiments, we train the Stanford parser with our developed treebank without any changes on its annotation granularity (Model 1), and we consider it as the baseline. To further study the effect of each annotation dimension, we do our experiments in three steps such that in each step only one dimension is focused on. The fine- vs coarse-grained granularities at

Table 6.8: Parsing results for applying different annotation dimensions using gold POS tags where *L* stands for 'Lexical Item', *T* for 'POS Tag', *C* for 'Constituent Label' (The differences are statistically significant according to 2-tailed *t*-test ($p < 0.01$))

Model	**Annotation Dimension**			**F-measure**	**Precision**	**Recall**	**EM**	**LA**
	L	T	C					
Model 1	Word	Fine	Fine	59.42	59.44	59.40	4.992	85.65
Model 2	Word	Coarse	Fine	47.38	47.39	47.37	4.046	80.64
Model 3	Word	Fine	Coarse	61.26	61.26	61.25	8.35	86.34
Model 4	Word	Coarse	Coarse	48.25	48.27	48.23	5.811	81.48

each of the annotation dimensions result in four possible configurations of which the obtained results are reported in Table 6.8. The differences between the performance of the models are statistically significant according to 2-tailed *t*-test ($p < 0.01$). The effect of annotation granularity at the lexical level is studied further in Chapter 7.

To find out the effect of detailed morpho-syntactic information on parsing, we build a model that its POS tags are coarse-grained but its lexical items and constituent labels are fine-grained (Model 2). Comparing this model with the baseline, there is a significant drop on the performance of the parser which indicates that the detailed morpho-syntactic information in the POS tags plays a very important role on parsing. Even though fine-grained POS tags increase the complexity of the tagging task as observed in the accuracy of the taggers, they have a positive impact on the parsing performance because of using detailed information for defining the dependencies.

To study the effect of the HPSG-based annotation on parsing, we build a model in which the constituent labels are coarse-grained, but the lexical items and the POS tags are fine-grained (Model 3). The results indicate that identifying the type of head-daughter dependencies is a hard task for a statistical parser, since the number of constituent labels in the HPSG-based treebank is higher than the simple labels in a treebank developed based on PSG. This might be the main reason that parsers trained with the data based on PSG have a higher performance and wider usage for NLP applications than the ones trained with the data based on the HPSG-based annotation. However, it must be emphasized that coarse-grained annotation loses valuable linguistic information that resembles the lexical semantic information. Simplifying the parsing complexity by using coarse-grained constituent labels is useful for applications that require simple syntactic analyses of sentences; whereas the semantic information modeled in the HPSG-based data set is valuable for applications, such as semantic role labeling, which need a deep analysis.

In Model 4, both of the POS tags and the constituent labels are coarse-grained and only the lexical items are fine-grained to study the effect of the interaction between POS tags and constituent labels. As can be seen, this model does not beat the baseline which indicates that there is a positive interaction between the POS tags and the constituent labels. The obtained results determine that losing detailed information has a negative impact on parsing.

6.7 Summary

In this chapter, we mainly introduced the NLP tools and the data required for our study. In the first run of our experiments, we assumed a real application by training the Stanford POS tagger with the Bijankhan Corpus and training the constituency parsers with our developed treebank. The obtained results showed the negative impact on parsing caused by the POS tagger. Two other parsing scenarios were proposed among which the one that used the gold POS tags was employed for the rest of the experiments to reduce the interfering of POS tagging on parsing. We experimentally found that the Stanford parser, which is a robust constituency parser, is a good choice for our further studies and experiments. In the next step of parsing, the constituency treebank was converted into its parallel dependency-based treebank to train and evaluate two dependency parsers. We use the dependency parsing in Chapter 8. Furthermore, we studied the impact of the treebank annotation granularity on parsing to demonstrate that the performance of the parser depends on other factors in addition to the size of the training data. To reduce the data sparsity problem, we propose two machine learning methods discussed in Chapters 7 and 8.

Chapter 7

Class-based Parsing

7.1 Introduction

Statistical parsers are very sensitive to the data they are trained with. The main problem with these parsers is that they require a large amount of data to create an accurate grammar model, therefore it is very difficult to build an accurate model confronted with sparseness of data. Additionally, it is very likely to encounter unknown words while parsing in a real application. Word clustering has caught attention in NLP to represent more coarse-grained concepts in place of the words themselves. In this approach, similar words are grouped together in one cluster according to a similarity metric. The words are clustered in an off-line process based on their occurrence in an unannotated corpus using an unsupervised method. In our study, we use the Brown word clustering algorithm (Brown et al., 1992), which is a bottom-up hard clustering algorithm, to reduce the data sparsity problem. In this chapter, we describe this clustering method and study its impact on reducing the data sparsity problem at the lexical level in parsing. We further investigate the effect of the annotation granularity on class-based parsing.

This chapter contains seven sections. Section 7.2 proposes class-based parsing to reduce the data sparsity problem. The Brown word clustering algorithm is then described in Section 7.3. Section 7.4 proposes an extended version of the normal Brown word clustering to deal with homographs. The performance of our proposed class-based model is compared with the normal Brown word clustering and the word-based model as the baselines in Section 7.5. Furthermore, other aspects of the annotation granularity that affect parsing performance are studied and evaluated in Section 7.6. This chapter is summarized in Section 7.7.

The content of this chapter is mainly based on the following paper:

- Ghayoomi, Masood (2012) "Word clustering for Persian statistical parsing". In *Advances in Natural Language Processing*, editors Hitoshi Isahara and Kyoko Kanzaki, volume 7614 of *Lecture Notes in Computer Science: JapTAL '12: Proceedings of the 8th International Conference on Advances in Natural Language Processing*, pp: 126–137. Springer Berlin Heidelberg.
- Ghayoomi, Masood and Omid Moradiannasab (2012) "The effect of treebank annotation granularity on parsing: A comparative study" In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, November 30–December 1, 2012, Lisbon, Portugal, pp: 109–114.

7.2 Properties of the Class-based Model

Rokach and Maimon (2005) identified two major clustering methods: (a) hierarchical clustering in which there is taxonomy on the clusters by using bottom-up or top-down clustering techniques, such as hierarchical agglomerative clustering and hierarchical divisive clustering, and (b) partitioning clustering, which is a flat clustering method with or without any overlap on the clustered data, such as the *k*-means clustering algorithm (MacQueen, 1967).

Brown et al. (1992) pioneered the use of the word clustering method and its application for language modeling. They used a hierarchical bottom-up approach for this goal. Later on, word clustering was widely used in various NLP applications including parsing (Koo et al., 2008; Candito and Crabbé, 2009; Candito and Seddah, 2010; Candito et al., 2011), word sense disambiguation (Li, 2002), automatic thesaurus generation (Hodge and Austin, 2002), machine translation (Uszkoreit and Brants, 2008), sentence retrieval (Momtazi and Klakow, 2009), named-entity tagging (Miller et al., 2004), language model adaptation (Kneser and Peters, 1997), speech recognition (Samuelsson and Reichl, 1999), query expansion (Aono and Doi, 2005), and text categorization (Chen et al., 2004).

Word clustering has advantages and disadvantages. One of the advantages of word clustering is that it reduces the data sparsity problem. Hence, if a word in the training data is not seen but its mapped class is met, then the performance of the system will not drop due to the sparseness of data and encountering unknown words (the out of the vocabulary problem). This approach is very effective, especially when the genre and domain of data change. Another advantage of word clustering is its flexibility to capture different features like semantic or syntactic properties of words by employing different word clustering algorithms. Since our aim is to group words with similar syntactic and semantic behavior, this flexibility gives us the opportunity to choose an algorithm which captures the syntactic and semantic similarities of words to be used for parsing. Assuming that the word clustering algorithm clusteres the words of a text accurately, it is obvious that there is a clear relationship between the words belonging to the same cluster. Example 7.1 shows instances of word clusters created by the Brown algorithm (Brown et al., 1992) for Persian.

(7.1)

CLUSTER-I	CLUSTER-II	CLUSTER-III
پرخطرترین	پاکیزگی	فرموده‌اید
/porxatartarin/	/pākizegi/	/farmudeʔid/
‘the most dangerous’	‘cleanness’	‘have prescribed’
شمالی‌ترین	بستنی	کرده‌است
/šomālitarin/	/bastani/	/kardeast/
‘the most Northern’	‘ice-cream’	‘has done’
ضعیفترین	زیبایی	کرده‌اند
/zaʔiftarin/	/zibāyi/	/kardeand/
‘the weakest’	‘beauty’	‘have done’
...	...	...

The word clusters help us to find the set of terms that are syntactically and semantically related to each other. If only one of the words from a cluster appears in the training data, the statistical parser can parse the input sentences which contain other words of the same cluster, even though these words do not exist in the training data. For example, if the word ‘پرخطرترین’ /porxatartarin/ ‘the most dangerous’ has been seen in the training data and this word creates a noun phrase with the word ‘مسیر’ /masir/ ‘path’ as ‘پرخطرترین مسیر’ /porxatartarin masir/ ‘the most dangerous path’, the class-based model is able to parse sentences that contain the word ‘شمالی‌ترین’ /šomālitarin/ ‘the most Northern’ which is unseen in the training data but belongs to the same cluster of ‘پرخطرترین’ /porxatartarin/ ‘the most dangerous’. Consequently, it can combine with the term ‘مسیر’ /masir/ ‘path’ to create a constituent like ‘شمالی‌ترین مسیر’ /šomālitarin masir/ ‘the most Northern path’.

The disadvantage of word clustering is that different syntactic behaviors of homographs are not distinguished, and they are grouped in one cluster. This problem might have a negative effect on applications like parsing. To reduce the problem of mis-clustering homographs when using a hard clustering approach, we extend the normal Brown word clustering algorithm described in Section 7.4 to recognize homographs distinctly. Furthermore, a soft clustering approach may sound a good solution to overcome this problem as well, but Dhillon et al. (2002) showed that the overall performance of hard clustering is still better than soft clustering.

When the clustering approach is used for the parsing application, two parsing models, namely word- and class-based parsing, can be created. In word-based parsing, the parser is trained with the treebank containing the words with their associated POS tags and the syntactic annotations. Class-based parsing is performed in two steps. In the first step, all lexical items of a corpus are clustered into a set of classes. Next, the words in the treebank are mapped to their corresponding clusters. The result is a treebank that contains clusters of words rather than the actual words. In the next step, the parser is trained with the treebank containing word clusters. In the following section, the Brown word clustering algorithm is described in detail.

7.3 Word Clustering with the Brown Algorithm

The Brown word clustering (Brown et al., 1992) is a hierarchical bottom-up algorithm which uses Average Mutual Information (AMI) of adjacent clusters to merge cluster pairs. When AMI is used, the algorithm examines contextual information to find similar words and put them in the same cluster. To this end, a set of word bigrams, $f(w, w')$, from the input corpus is required, where $f(w, w')$ is the number of times the word w' is seen after the word w. Both w and w' are assumed to come from a common vocabulary. When this algorithm is used for word clustering, different words seen in the same contexts will merge, because the appearance of the words in the same context shows that these target words can be replaced by each other and they are assigned to the same cluster as a result (Morita et al., 2004). Samples can be found in Example 7.1.

As mentioned, the Brown algorithm (Brown et al., 1992) uses AMI as the similarity measure. Mutual Information (MI) computes the amount of information that two words share. The MI of the two adjacent clusters $(C_w, C_{w'})$ is calculated in the equation (7.1):

$$MI(C_w, C_{w'}) = \log \frac{P(C_w, C_{w'})}{P(C_w) * P(C_{w'})} \tag{7.1}$$

If w' follows w less often than we expect on the basis of their independent frequencies, then MI is negative. If w' follows w more often than we expect, then MI is positive (Brown et al., 1992). Algorithm 3 shows the Brown word clustering algorithm in more detail.

As shown in the algorithm, clusters are initialized with a single term in each cluster. Then, in each iteration, the best cluster pair, which offers a minimum decrement in AMI, combines together. The process continues for $V - K$ iterations, where V is the number of terms and K is the predefined number

Algorithm 3 Brown word clustering algorithm

```
Initial Mapping: Put a single word in each cluster
                 Compute the initial AMI of the collection
repeat
  Merge the pair of clusters which has the minimum decrement in AMI
  Compute the AMI of the new collection
until reach the predefined number of clusters
repeat
  Move each word to the cluster that offer the highest AMI
until no change is observed in AMI
```

of clusters. In the final step of the iterative process, all words are temporarily moved from one cluster to the other cluster one by one, and AMI is recalculated. If this reassignment increases AMI, then the word will move to a cluster which offers the highest AMI. The algorithm stops when no additional increment in AMI is observed.

7.4 Extension of the Brown Word Clustering

As described in the previous section, the Brown algorithm originally uses the word bigrams from a raw corpus for clustering (thereafter we call it Model A). The output of this clustering is hard, i. e. each word is assigned to only one cluster. The advantage of using clustering is that it reduces the data sparsity problem, which consequently has a positive impact on statistical parsing. However, the main shortcoming of hard clustering is that each lexical item is restricted to one class, which is not ideal for homographs. This problem is more pronounced for Persian text processing, since short vowels are written rarely and this property makes the number of homographs relatively high. Bijankhan et al. (2011) defined syntactic patterns to distinguish Persian homographs.

To reduce the problem of mis-clustering homographs, we propose using the main POS tag of the words to disambiguate a large portion of homographs for clustering (thereafter we call it Model B). As an example, the word ‘شوم’ can be either pronounced /šum/ as an adjective which means ‘evil’ or /šavam/ as a verb in first person singular which means ‘become’. The word ‘برداشت’ /bardāšt/ is another example which is both a homograph and a homophone. It can be a noun which means ‘understanding’ or a verb in third person singular which means ‘picked up’. The normal word clustering treats these homographs equally, and they are assigned to only one cluster as a result. While in our extension, the main POS tag of the word is used as additional lexical information for clustering. As a result, the homographs which have different POS tags are assigned to different clusters. To prepare the input corpus for the extended model as the input data of the Brown algorithm, the POS tag of a word is attached to the word with a hyphen, such as: ‘شوم-ADJ’ and ‘شوم-V’, or ‘برداشت-N’ and ‘برداشت-V’.

7.5 Evaluation

7.5.1 Experimental Setup

To evaluate the performance of parsing in the class-based model, we must cluster lexical items by the Brown word clustering algorithm described in Section 7.3. To this end, we use the SRILM toolkit (Stolcke, 2002) which contains the implementation of the Brown algorithm. This toolkit is developed to build statistical language models.

For class-based models, the treebank is converted in such a way that the words of the treebank are mapped to the associated clusters in the proposed class-based parsing models. We use 10-fold cross-validation to evaluate our models and study the clustering impact on parsing performance.

7.5.2 Results and Discussion

Since the Brown algorithm requires a pre-defined number of clusters, we perform our experiments on 100, 500, 1000, and 1500 clusters of the vocabulary terms from the Bijankhan Corpus (Bijankhan, 2004) and PLDB (Assi, 2005). Table 7.1 represents the performance of the class-based parsing using Model B with different numbers of clusters.

Table 7.1: Performance of the Stanford parser for extended class-based parsing (Model B)

Number of Clusters	**F-Measure**
100	63.70
500	67.25
1000	67.36
1500	67.00

As can be see in the table, the performance of the parser is not very sensitive to the number of clusters, which demonstrates that fine-tuning the number of clusters is unnecessary, and we can still obtain a reasonable result when different number of clusters is used. Nonetheless, according to the experimental results, the best performance is achieved by clustering all vocabulary terms into 1000 clusters. We use the clustered lexical items of the 1000 clusters for class-based models.

Table 7.2 compares the results of the class-based parsing (Model A and Model B) with word-based parsing (the baseline). As shown in the table, the

Table 7.2: Performance of the Stanford parser for word- and class-based parsing

Model	**F-Measure**	**Precision**	**Recall**	**EM**	**LA**
Word	59.42	59.44	59.40	4.992	85.65
Class (A)	66.91	67.02	66.79	7.175	87.67
Class (B)	67.36	67.46	67.26	6.966	87.74

class-based models outperform the baseline. The differences between the performance of the class-based models and the word-based model are statistically significant according to the 2-tailed t-test ($p < 0.01$). This result indicates that even though the class-based approach generalizes the word representation, it reduces the data sparsity problem at the lexical level and it has a positive impact on the performance of statistical parsing. Moreover, the proposed extension of clustering algorithm (Model B) outperforms Model A. Based on the results, resolving the problem of clustering homographs does have a positive impact on parsing. The improvement in Model B compared with Model A is statistically significant according to the 2-tailed t-test ($p < 0.05$).[1] Based on the results summarized in Table 7.2, we can see the same behavior in the precision and recall of the models; i. e. precision and recall of the class-based models are higher than the word-based model and Model B performs the best. We use our extended class-based model using 1000 clusters, which performs the best, for the rest of our study.

In terms of the vocabulary size used in the clustering models, 64,820 terms are used for Model A, and 67,877 terms for Model B. This shows that around 3,057 more terms are added to the vocabulary of Model B which obviously indicates that our proposed extended model has a softer clustering for homographs.

7.6 Other Aspects of Annotation Granularity

7.6.1 Experimental Setup

Section 6.5 described three possible annotation dimensions for parsing: lexical item, POS tag, and constituent label. In Section 6.6.2, we studied the impact of fine- and coarse-grained annotation of POS tags and constituent labels on word-based parsing (fine-grained lexical items). In the following, we further study the impact of POS tags and constituent labels annotation granularity using class-based parsing (coarse-grained lexical items). The procedure for developing coarse-grained POS tags and constituent labels of the data for our experiments is described in Section 6.6.2.

[1] We further employed the extension of the Brown word clustering on the Bulgarian language for the parsing application and obtained similar results (Ghayoomi et al., 2014).

Table 7.3: Parsing results for applying different annotation dimensions using gold POS tags where *L* stands for 'Lexical Item', *T* for 'POS Tag', *C* for 'Constituent Label' (The differences are statistically significant according to 2-tailed *t*-test ($p < 0.01$))

Model	**Annotation Dimension**			**F-measure**	**Precision**	**Recall**	**LA**	**EM**
	L	T	C					
Model 1	Word	Fine	Fine	59.42	59.44	59.40	4.992	85.65
Model 2	Word	Coarse	Fine	47.38	47.39	47.37	4.046	80.64
Model 3	Word	Fine	Coarse	61.26	61.26	61.25	8.35	86.34
Model 4	Word	Coarse	Coarse	48.25	48.27	48.23	5.811	81.48
Model 5	Class	Fine	Fine	67.36	67.46	67.26	6.966	87.74
Model 6	Class	Coarse	Fine	61.31	61.46	61.16	7.166	85.16
Model 7	Class	Fine	Coarse	68.89	68.97	68.81	11.76	88.35
Model 8	Class	Coarse	Coarse	62.56	62.73	62.40	9.898	85.82

7.6.2 Results and Discussion

Section 6.6.2 reported the results of the annotation granularity using word-based parsing (Models 1 to 4). In this section, we report the impact of coarse-grained lexical items on parsing (Models 5 to 8) and compare the results with the word-based parsing. Table 7.3 summarizes the obtained results of all models.

To determine the effect of the data sparsity problem at the lexical level, we examine a class-based model with fine-grained annotations of POS tags and constituent labels (Model 5). When comparing the results with the baseline (Model 1), the class-based parsing outperforms the word-based model. The achieved result indicates the negative impact of the data sparsity problem and the superiority of coarse-grained representation of lexical items on parsing. In Model 6, the lexical items and the POS tags are coarse-grained and the constituent labels are fine-grained to study the effect of the HPSG-based annotation without the impact of the morpho-syntactic information and the data sparsity problem. Similar to the results obtained between Models 1 and 2, the negative impact of losing morpho-syntactic information of POS tags on parsing is obvious when Models 5 and 6 are compared. In Model 7, the lexical items and the constituent labels are coarse-grained, while the POS tags are fine-grained. This model is built to study the effect of available morpho-syntactic information in case there is a reduction on the sparseness of data without the effect of the HPSG-based annotation. From the results of Models 2, 3, and 5, we infer that the detailed morpho-syntactic information in the POS tags, the coarse-grained representation of the constituent labels, and the class-based parsing have positive impacts on parsing. The impact of these three variables are represented together in Model 7, which outperforms all of the models. In contrast, Model 2, which has the opposite configuration, performs the worst. Finally, in Model 8, the coarse-grained representation of the information at the three dimensions is studied. The comparison of Models 1 and 8 indicates that better results are obtained when there is a coarse-grained representation of the linguistic knowledge, but higher results are obtained when richer POS tags are used, as is done in Model 7. The comparison of Models 4 and 8 indicates that reducing the data sparsity problem results in a high performance. The comparison of Models 6 and 8 indicates the negative impact of the HPSG-based annotation on parsing, since it is a hard task for the parser to determine the type of the dependency relations when the coarse-grained representation of the POS tags is used. While a better performance is obtained when more fine-grained syntactic categories is available as determined in Model 5.

Based on the results reported in Table 7.3, we can study the effect of each annotation dimension on all possible configurations. The comparison of Models 5 and 1, Models 6 and 2, Models 7 and 3, and Models 8 and 4 shows that the first model in each pair beats out the second model. This result indicates that the class-based model always outperforms the word-based model, disregarding the granularity of the POS tags and the constituent labels. There can be a similar study on the effect of POS tag annotation by comparing Models 1 and 2, Models 3 and 4, Models 5 and 6, and Models 7 and 8. In all of the models, the first model in each pair beats out the second model. This result indicates the superiority of using fine-grained annotation for POS tags in parsing, disregarding the granularity of the lexical items and constituent labels. To study the impact of the constituent labels, Models 2 and 4, Models 3 and 1, Models 7 and 5, and Models 8 and 6 are compared. In all of the models, the first model in each pair beats out the second model. This result shows that the coarse-grained annotation of the constituent labels always result in a higher parsing performance, disregarding the granularity of lexical items and the POS tags. It must be mentioned that the differences between the performance of all eight models are statistically significant according to the 2-tailed *t*-test ($p < 0.01$).

7.7 Summary

In this chapter, we proposed a word clustering approach for parsing as an alternative method for reducing the data sparsity problem at the lexical level. To this end, we employed the Brown word clustering and its extended version to handle homographs, and then trained the parsers with the clustered data. We proved experimentally that using a clustering method to reduce the data sparsity problem has a positive impact on parsing to outperform the word-based model as the baseline. Additionally, we showed that the proposed extended model performs the best.

We further studied the effect of annotation granularity on parsing from three dimensions on Persian. Comparing the results determined that coarse-grained representation of the lexical items has a positive impact on parsing. Furthermore, the detailed morpho-syntactic information of POS tags plays an important role in parsing and losing this information drops the performance. Moreover, determining the appropriate label for constituents reduces the performance of the parser due to increasing the complexity for determining the correct type of head-daughter dependencies.

Chapter 8

Active Learning for Treebank Enlargement

8.1 Introduction

Machine learning is one of the sub-fields of artificial intelligence which tries to simulate the intelligent abilities of humans in machines. The learning process can be unsupervised or supervised. In the former learning model, the learner uncovers the hidden regularities in the data and clusters the unannotated data based on its similarity without any preliminary knowledge. The latter learning model is a two-step learning process in which it first infers patterns inductively from the training data, and then it makes predictions on unseen events based on the learnt patterns.

In the previous chapter, we proposed an unsupervised approach to reduce the data sparsity problem at the lexical level for the parsing application. In this chapter, we propose models to relatively reduce the data sparsity problem at the syntactic construction level. To this end, we use active learning as one of the supervised machine learning methods to select new informative sentences to be added to the treebank and to enlarge the treebank. Consequently, we can improve parsing performance and minimize the amount of human effort required for data annotation. To reach the goal, we propose several models according to two sampling methods: uncertainty sampling, and query-by-committee.

This chapter contains six sections. In Section 8.2, we first describe active learning. The active learning scenarios, and two of the most well-known methods used for selecting informative samples from the data pool are introduced. In Section 8.3, the previous studies on using this learning method for parsing are discussed. Our proposed sampling methods and the baselines are described in

Section 8.4. The evaluation results of the proposed active learning models are discussed in Section 8.5. Section 8.6 summarizes the chapter.

The content of this chapter is mainly based on the following paper:

- Ghayoomi, Masood and Jonas Kuhn (2013) "Sampling methods in active learning for treebanking". In *Proceedings of the 12th International Workshop on Treebanks and Linguistic Theories*, December 13–14, 2013; Sofia, Bulgaria, pp: 49–60.

8.2 Active Learning

Supervised learning models require annotated data, but the development of such data is expensive, time-consuming, and tedious. Active learning is one of the supervised machine learning methods which creates annotated data with the help of human intervention in an iterative process (Thompson et al., 2003; Busser and Morante, 2005). The motivation behind active learning is to maximize the performance of a system with minimum amount of data to be annotated. The consequence is minimizing the amount of human effort required to annotate new data. In each iteration of the active learning process, the learner is trained with the training data, and then a small subset of the unannotated data is selected and handed to the oracle to be annotated. Finally, the newly annotated data is added to the initial learner's training data, and the learner is retrained with the augmented data. This process continues in a loop until it terminates.

In the followings, the learning scenarios and the well-known sampling methods frequently used in active learning for NLP applications as well as the stopping criteria to stop the active learning process are briefly explained. Further detail can be found in Settles (2009, 2012) which gives a comprehensive explanation of active learning, in general.

8.2.1 Learning Scenarios

There are two main learning scenarios in active learning: (a) stream-based, and (b) pool-based. In the former scenario, the learner takes one sample at a time from the data and tries to decide whether to select and hand it to the oracle to be annotated or to disregard it. In the latter scenario, first the learner takes all the samples from a data pool, and then it ranks them based on a sampling criterion in ascending order. Finally, the learner selects the top k samples from this ranked list in each iteration and hands the selected samples to the oracle for annotation.

In the pool-based scenario, the number of informative samples to be selected from the pool of unannotated data is fixed and does not change, and the selection is done based on ranking. Whereas in the stream-based scenario, the selection is done based on a threshold, and it immediately asks the oracle for the annotation of the selected sample (Baram et al., 2004).

8.2.2 Sample Selection Methods

Uncertainty Sampling

Lewis and Gale (1994) introduced 'uncertainty sampling', and it has become the most well-known and simple sampling method. In this method, the active learner hands the most uncertain samples to the oracle. When the learner annotates unannotated data, a distinction should be made between *uninformative* and *informative* samples. Uninformative samples are the ones in which the learner has the highest confidence and certainty on the provided annotation. These samples are not very useful for the learner, and it is not necessary to ask the oracle to annotate them since they contain redundant information. The informative samples are the ones for which the learner does not have a good solution with high confidence, and therefore they are more interesting for the active learner.

The border line between uninformative and informative samples can be defined by a confidence score that the learner returns for the proposed annotation. There are different ways to use the confidence score. Among them, 'entropy' (Shannon, 1948) is the most popular sampling criterion used for information retrieval (Zhang and Chen, 2002) and NLP applications, such as parsing (Hwa, 2000, 2001, 2004; Baldridge and Osborne, 2003), word sense disambiguation (Chen et al., 2006a; Zhu and Hovy, 2007), and text classification Zhu et al. (2008b). To select the informative samples according to the entropy-based sampling for parsing, entropy of all possible annotations for each sample is first computed. Next, samples which have high entropy are selected as informative samples. When entropy is high, the prediction score of the learner for these samples is low, as a result the samples with high entropy are the best candidates to be annotated by the oracle.

One property of uncertainty sampling is that only one learner is needed for sampling. It is also possible to employ more than one learner for sampling which is described in the following section.

Query-by-Committee

Seung et al. (1992) and Freund et al. (1997) proposed 'query-by-committee' as another sampling method widely used for different applications. In this method, more than one learner is used such that each learner predicts an annotation for the unannotated data. Among the samples, those which have the highest degree of disagreement in the predicted annotation between the committee of the learners are selected as informative samples, and they are handed to the oracle for annotation.

8.2.3 Stopping Criteria

As mentioned, active learning is an iterative process that should be stopped at some point. Depending on the application of active learning, one can stop the active learning process when a desired amount of data is annotated. It is also possible to use confidence measures to stop it in an ideal point when the confidence score of the learner for annotation is higher than the pre-defined threshold. The other alternative to stop it is to use the performance of the learner as a criterion. In this method, the learning process stops when the learner has reached its maximum performance and annotating more data does not have any impact on the performance of the learner. In our current proposed models, we do not use any stopping criteria since it is out of the scope of our study. However, this feature, which is taken into consideration in the literature by researchers like Zhu and Wang (2008), Vlachos (2008), Laws and Schütze (2008), and Ghayoomi (2010) among others, can be considered as future work when the model is used in a real application.

8.3 Previous Studies of Using Active Learning for Parsing

Active learning is used for reducing the amount of data to be annotated while obtaining a performance similar to the performance of a learner trained with the complete set of annotated data. This leads to reduction of human effort to annotate data. Active learning can be used for various NLP applications, including constituency or dependency parsing. In the followings, we briefly describe the sketch of the previous studies on using active learning for constituency parsing and treebanking. The main idea behind using active learning for parsing in the studies is to find informative sentences with minimum redundancy in the training data.

Ratnaparkhi (1999) proposed an active learning model which used a maximum entropy parser. This parser selects the parse trees with maximum entropy probability from a set of derived parsed trees. In the maximum entropy framework, features are required as evidences to build the model. These features provide the contextual information represented as chunks. The less specific contextual information is more interesting in this active learning model to "provide reliable probability estimates when the words in the history are rare" (Ratnaparkhi, 1999). The advantages of this sampling method are reduction of the impact of sentence length on selection and building an independent domain model.

Hwa (2000, 2001, 2004) proposed a sampling method for inducing 'probabilistic lexicalized tree insertion grammar'. In each iteration of the model, the entropy of each sentence is calculated based on the probability scores of its candidate parse trees. If a grammar is certain about the structure of a sentence, then one parse tree will be assigned a high probability score and the rest low scores, which results in a low entropy. Whereas, for uncertain sentences, all candidate parse trees will have a uniform probability score and a high entropy, as a result. The Tree Entropy (TE) of sentence s is calculated according to the equation (8.1).

$$\begin{aligned} TE(s,G) &= -\sum_{v\in V} p(v)log_2(p(v)) \\ &= -\sum_{v\in V} \frac{P(v|G)}{P(s|G)} log_2(\frac{P(v|G)}{P(s|G)}) \end{aligned} \tag{8.1}$$

where V $(v \in V)$ is the set of possible parses of s. $P(v|G)$ is the probability score of parse tree v which is provided by the parser using the built-in grammar G. $P(s|G)$ is the sum of probabilities of its parses as in the equation (8.2).

$$P(s|G) = \sum_{v\in V} P(v|G) \tag{8.2}$$

Steedman et al. (2003) proposed a co-training model in which two different parsers are employed for sampling without any manual annotation. In each iteration of the model, a small set of sentences is pulled out from the data pool and stored in a cache. Next, the parsers parse the sentences in the cache. After that, a subset of the parsed sentences is selected and added to the training data. The selected data is the output of one parser, which is added to the training data of the other parser. During the selection step, one parser first acts as a teacher and the other parser as a student, and then the roles are reversed. The most important feature in this model is the selection process, which is based on the accuracy score rate. To this end, two scoring functions are defined: (a) a scoring function based on the F-measure of the analysis against the gold data, and (b) a scoring function based on the conditional probability of the parser. Three sampling methods are exploited in the model: (a) defining a score as a threshold to select the most accurate analyses, (b) computing the difference between the score of the teacher and the student to choose the candidates in which the teacher is more accurate, and (c) finding the intersection between the n percent highest scores of the teacher and the n percent lowest scores of the student for the same sentence to select sentences which are accurately parsed by the teacher and incorrectly by the student.

Baldridge and Osborne (2003), Baldridge and Osborne (2008), and Osborne and Baldridge (2004) described a method for active learning of HPSG parse selection. They used the entropy-based uncertainty sampling, a query-by-committee sampling method between log-linear and perceptron algorithm (Rosenblatt, 1957) as the learners, and a combined selection method that takes the intersection of entropy- and disagreement-based models. Features are required to build the models. Two feature selection methods are exploited: (a) selecting features from derivation trees, and (b) extracting n-gram features from flattened derivation trees treated as a sequence of rule names. Baldridge and Osborne (2004) further studied the model to measure the annotation cost to evaluate the effectiveness of different experiments of active learning conditions.

Hughes et al. (2005) described a system for selecting CCG parses through an interactive correction process. In this model, human annotators have an interaction with the system such that they add constraints to the parser, and the parser returns the most probable parse results which satisfy all constraints. The informative samples are selected via a pool-based active learning process in a query-by-committee model.

Lynn et al. (2012) built a query-by-committee model for developing the Irish dependency treebank. In their model, the disagreement between a committee of two parsers is considered as the sampling metric. The Malt and Mate dependency parsers are employed in their models.

8.4 Active Learning for Persian Treebanking

In this section, we describe our sampling methods to select the most informative sentences from a data pool to enrich our treebank and to reduce the sparseness of the syntactic constructions. In our active learning application for treebanking, the learner is a parser that learns the grammar of the Persian language. To create the models, we employ the two introduced sampling scenarios to find the best strategy for our purpose. In this study, instead of enlarging the current treebank, we simulate the active learning process, and we use our developed treebank to be able to implement all models proposed and to compare them with each other. The best model can be further used for a real application.

8.4.1 Basic Sampling Methods

Random Sampling

Random sampling is a basic method for selecting samples without using any intelligence for the selection. In this sampling method, k sentences are selected randomly from the data pool. The method which creates a model according to the data of this sampling method is called passive learning. Passive learning is usually considered as a baseline for active learning.

Sentence Length Sampling

We introduce a naive sampling method as another simple baseline. This sampling method uses sentence length as a criterion to select complex sentences that contain more lexical items. The longest sentences are assumed to be the most informative samples. In this passive learning method, first the sentences in the data pool are ranked in descending order according to their length, and then k sentences are selected from the top of the list.

8.4.2 Entropy-based Sampling

We use the entropy-based uncertainty sampling method proposed by Hwa (2000), which was described in Section 8.3, as a metric to measure the uncertainty of a sentence in a grammar according to the parse results. Based on this criterion, we propose three different models. In the first two models, only the tree entropy is used as the sampling criterion. In the third model, we exploit the two models and use a meta-ranker on top of the models to select the informative samples.

Model 1: Word-Entropy

In Model 1, the actual sentences in the treebank are used for training and testing the active learning model. Since we have a small treebank, we suffer from the data sparsity problem at both the lexical and syntactic construction levels. In each iteration of the active learning process, k informative sentences which have high entropy and contain informative lexical items and syntactic constructions for the grammar model are selected from the data pool.

Algorithm 4 displays the steps for selecting uncertain samples in the uncertainty models. In our active learning application, which is treebanking, the learner is a parser that learns the grammar of the Persian language.

Algorithm 4 Active learning with entropy-based uncertainty sampling

```
Input: Seed data S from the Persian HPSG-based treebank,
       Pool of unlabeled samples U
repeat
  Step1: Use S to train the parser P
  Step2: Use P to parse U, and extract $n$-best parse trees for each sentence
  Step3: Compute tree entropy TE of each sentence based on the probability score
         of the $n$-best parse trees
  Step4: Sort the sentences based on their TE score in descending order
  Step5: Select top K samples from the sorted parsed trees
  Step6: Augment S with K samples, and remove K from U
until the stopping criterion is met
```

Model 2: Class-Entropy

In Model 1, two sparsity levels (at the lexical and syntactic construction levels) are taken into consideration for sampling simultaneously. As a result, the sentences which are the most informative ones at the both levels are selected. To minimize the data sparsity problem at the lexical level and to select sentences with informative syntactic constructions, we use a class-based parsing model (Model 2). In this model, we use the extended Brown word clustering algorithm described in Section 7.4. After mapping the words of the treebank into their corresponding clusters, the active learning process starts as demonstrated in Algorithm 4. In this model, the tree entropy is calculated based on the class-based parsing. Next, k informative sentences with high entropy are selected from the data pool. The selected sentences are then reconverted into the word-based format and added to the treebank. To make the result of this model comparable with other models and have a fair comparison, the performance of the parser is evaluated according to the word-based data format. The advantage of the class-based parsing is that the data sparsity problem at the lexical level is reduced and it has a minimum impact on sampling.

Model 3: Combo-Entropy

Model 3 is a combination of Models 1 and 2. In this model, after using Algorithm 4 to find the informative samples in Models 1 and 2 separately, a meta-ranker on the top of the models are employed to re-rank sentences based on their average rank in both models. As an example, assume that sentence S is parsed with Models 1 and 2. The outputs of the models, S_w for the word-based model and S_c for the class-based model, obtain the ranks 12 and 4 in descending order of the outputs. Using the meta-ranker on the top of the models, the score 8 is assigned to S, and the position of S will be re-ranked based on the assigned new score. After re-ranking the sentences, the top k sentences from the re-ranked list are selected as the most informative samples. The advantage of this model is that the properties of Models 1 and 2 are taken into consideration for sampling.

8.4.3 Query-by-Committee

Model 4: Combo-Committee

In Model 4, a combination of word- and class-based parsing is used as represented in Algorithm 5. In this model, sentences are first parsed with either of the models individually, without using any criterion for sampling. Next, the samples which have the highest degree of disagreement between the learners are selected as the most informative samples. In this sampling, the output of the class-based parsing is converted into its equivalent word-based model. This conversion makes the output of the two parsing models comparable.

Algorithm 5 Active learning with query-by-committee sampling

```
Input: Seed data S from a treebank,
       Pool of unannotated samples U
repeat
   Step1: Train parsers P$_i$ ($i \in I$) with S
   Step2: Use P$_I$ to parse U
   Step3: Find samples that have the highest degree of disagreement between P$_I$
   Step4: Sort the samples based on a disagreement score in ascending order
   Step5: Select top K samples from the sorted samples
   Step6: Augment S with K samples, and remove K from U
until the stopping criterion is met
```

To find the disagreement between the learners, we use a method similar to the ‘F-complement’ sampling method proposed by Ngai and Yarowsky (2000). The equation in (8.3) computes the disagreement (D).

$$D = \frac{1}{2} \sum_{M_i, M_j \in K} (1 - F_1(M_i(e), M_j(e))) \tag{8.3}$$

where K is the committee model and M$_i$ and M$_j$ are the individual models in K, and F$_1$($M_i(e), M_j(e)$) is F$_1$ of M_i’s labeling of e relative to M_j’s evaluation of e.

To find the samples that the parsers disagree upon in our sampling method, the output of one parsing model is assumed as gold data (word-based parsing) and it is compared with the output of the other model assumed as guess data (class-based parsing). Next, we compute the F-measure based on the outputs of the two parsing models and sort sentences according to the F-measure in ascending order. The top k sentences, which obtain the lowest F-measure, are the most informative sentences in which the two parsing models disagree upon the provided analyses.

The main difference between the F-complement method (Ngai and Yarowsky, 2000) and our method is that in F-complement method two different constituency parsers are used as the committee members, whereas in our model a single parser with two different parsing models is employed. This makes it possible for us to find a simpler way to find the disagreements between the parsers.

Model 5: Dependency-Constituency-Committee

In Models 1–4, only a constituency parser is used for selecting the informative samples. In Model 5, we propose using constituency and dependency parsers such that first the sentences in the data pool are parsed by either of the parsers. To select samples in each iteration of this model, the constituency trees are converted into their equivalent dependency relations. Having the two sets of trees based on the dependencies makes it possible to consider one set of trees as gold data and the other set as guess data. After evaluating and sorting the results in ascending order of the unlabeled attachment score, k sentences which obtain the lowest score are selected as informative sentences in which the two parsing models disagree upon the provided analyses. To make the results comparable to other models, the dependency trees of the selected sentences are reconverted into their equivalent constituency trees to evaluate the performance based on constituency.

Model 6: Dependency-Combo-Committee

In Model 6, in an offline mode, the complete constituency treebank is converted into its equivalent dependency-based treebank. Next, the word-based dependency-based treebank is mapped to its equivalent class-based dependency treebank. The provided data sets are used for training a dependency parser. In Model 6, only a dependency parser is employed to construct the query-by-committee sampling method by involving word- and class-based parsing models.

In this model, sentences are first parsed by each individual word- or class-based parsing models. Next, the output of one parsing model (word-based parsing) is considered as gold data, and the output of the other parsing model (class-based parsing) is considered as guess data to compare the two parsing models. Similar to Model 5, the results are sorted in ascending order of the unlabeled attachment score. Finally, the top k sentences with the lowest unlabeled attachment score are selected as the informative sentences in which the two parsing models disagree upon the provided analyses. To make the results comparable to other models, the dependency trees of the selected sentences are reconverted into their equivalent constituency trees to evaluate the performance based on constituency.

8.4.4 Tree Similarity Sampling

Aside from the models proposed, we introduce a new criterion for selecting informative samples. In this sampling method, the degree of similarity between the syntactic trees of the parser's output and the input training data are computed. We hypothesize that the most informative samples in the data pool are those which have the lowest degree of similarity to the training data.

To measure the tree similarity, we use the tree kernel method proposed by Collins and Duffy (2001, 2002). To compute the tree similarity between two trees, first the two trees are decomposed into their corresponding subtrees to construct the feature vector. This vector is used for capturing the structural information. Next, the number of subtrees with the same vector are summed up. Since the length of sentences vary, this score should be normalized. For the normalization, the square of the product of the total number of substructure trees in the two trees is computed. The similarity degree is the ratio of the number of completely identical substructure trees to this normalization value. The similarity degree is a number between 0 and 1, where 0 indicates totally dissimilarity and 1 indicates 100% similarity between the two trees.
We use the similarity degree as a metric for selecting the informative sentences. As a result, all sentences in the active learning process are sorted in ascending order according to their tree similarity degree, and then the top k sentences which have the lowest similarity degree are selected as the informative samples. The set of selected sentences shares a minimum amount of similarity in their syntactic structures with the training data, and the sentences whose substructures are rare in the training data are the best candidates to be selected.

The following example illustrates how the tree similarity degree is computed. To measure the similarity degree of the tree in Example 8.1 to the tree in Example 8.2, the two trees must be first decomposed into their corresponding substructure trees, as represented in Figures 8.1 and 8.2. Next, the number of the substructure trees which are totally identical in the two tree sets are divided by the square of the product of the total number of substructure trees. In this example, out of the 5 recognized substructure trees in Figure 8.1, 3 of them are totally identical to the trees in Figure 8.2. Figure 8.3 displays the set of common identical substructure trees. The similarity degree of the tree in Example 8.1 to the tree in Example 8.2 is $\frac{3}{\sqrt{5\times5}} = 0.6$.

(8.1) a car

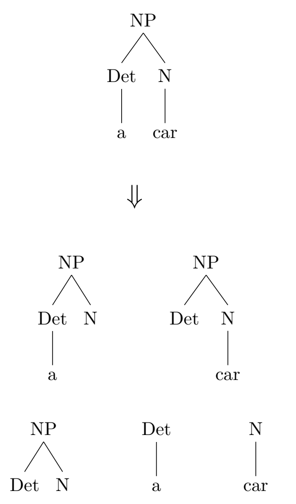


Figure 8.1: Substructure trees of Example 8.1

(8.2) a book

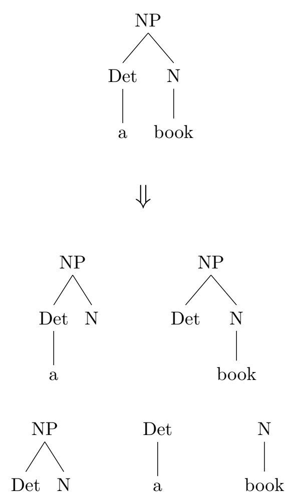


Figure 8.2: Substructure trees of Example 8.2

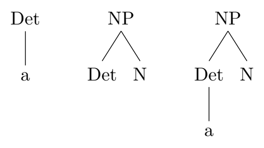


Figure 8.3: Common substructure trees between Examples 8.1 and 8.2

In our tree similarity metric, we use the words in the sentences, their corresponding POS tags, and the substructure trees as the features to compute the similarity degree. We propose two models for the tree similarity metric as the criteria to select informative samples.

Model 7: Word-Similarity

In Model 7, the words in the sentences, in addition to their corresponding POS tags, and the substructure trees derived from the parser's output are used in the feature set to compute the tree similarity degree between a tree as the output of the parser and the trees in the training data. Next, sentences are sorted in ascending order of their tree similarity degree, and the *k* samples with the lowest similarity degree are selected as informative samples.

Model 8: Class-Similarity

The drawback of Model 7 is that due to the data sparsity problem at the lexical level, the existence of unknown words in the data pool is unavoidable. To minimize this impact on selection, we use a class-based model. To this end, we use the extended Brown clustering algorithm described in Section 7.4. After computing the similarity degree of sentences, they are sorted in ascending order of their tree similarity degree, and k samples with the lowest similarity degree are selected as the informative samples. To make the results comparable with other models and to have a fair comparison, the output of the class-based parsing is reconverted into the word-based format, then the performance of the parser is evaluated.

8.5 Evaluation

8.5.1 Experimental Setup

We use the Stanford constituency parser (Klein and Manning, 2003) and the Mate dependency parser (Bohnet, 2009) in our proposed models. The HPSG-based treebank that we developed is used for training the constituency parser, and Algorithm 2 on page 147 is used for converting the constituency treebank into its parallel dependency-based treebank to train the dependency parser. Gold POS tags are used for parsing in our experiments to avoid the negative interfering of POS tags on parsing. The class-based approach described in Section 7.4 with cluster number 1000 is used for building the class-based models. To compute tree similarities, we use the FLinK software[1] developed by Pighin and Moschitti (2009).

The reported results are based on the simulation of active learning to choose the best approach for a real application to enlarge the current treebank. To this end, we divide the treebank into three subsets such that 20% of the data is considered as seed data to initialize the model, 10% as test data to evaluate the performance of the model in each iteration, and the remaining data is assumed as unannotated data to be annotated within the active learning process. 10-fold cross validation is used for evaluating the experiments. Since our experiments are based on the pool-based scenario, k informative sentences (k=10) are selected in each iteration. To compute tree entropy, the n-best parses (n=20) of each sentence are extracted in each iteration.

[1]`http://danielepighin.net/cms/software/flink`

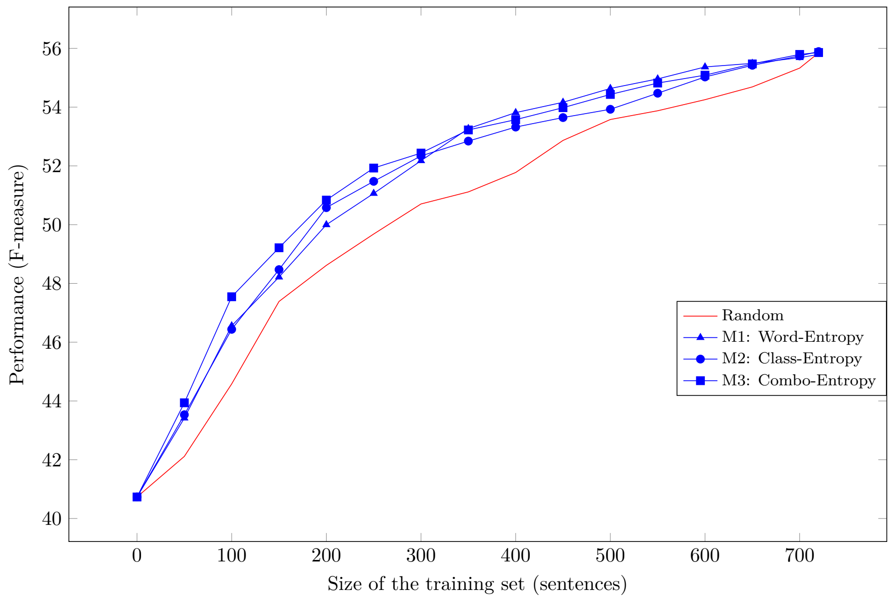


Figure 8.4: Learning curve of the learnt sentences in entropy-based uncertainty sampling models and the baseline

8.5.2 Results and Discussion

To make the results comparable, we first demonstrate the learning curves of the set of models with similar sampling criterion along with the baselines and then compare the results. Next, we compare the learning curves of the best models in each set and discuss the performance of the active learning models.

Figure 8.4 represents the learning curves of the entropy-based uncertainty sampling models (Models 1 to 3) and the results of random sampling as the baseline. To make the results more readable, instead of representing the result of each iteration, the average performance for each 5 iterations is shown in this figure. As can be seen in the curves, the three uncertainty sampling models have beaten random sampling, the baseline. This result indicates the superiority of using active learning models for our goal. In the early iterations, the results of Models 1 and 2 are comparable, but Model 3 outperforms the two models. In iteration 31, Model 1 beats Model 2, and later in iteration 35, it beats Model 3, and this model consistently performs better in the second half of the annotation process. Since it is expected to find the informative samples in early iterations of the active learning process, the first half of the annotation process is more important than the second half. Comparing Models 1 to 3 and the baseline, we can conclude that Model 3 that has a meta-ranker can be chosen as the best entropy-based uncertainty sampling for selecting informative samples to enlarge the treebank.

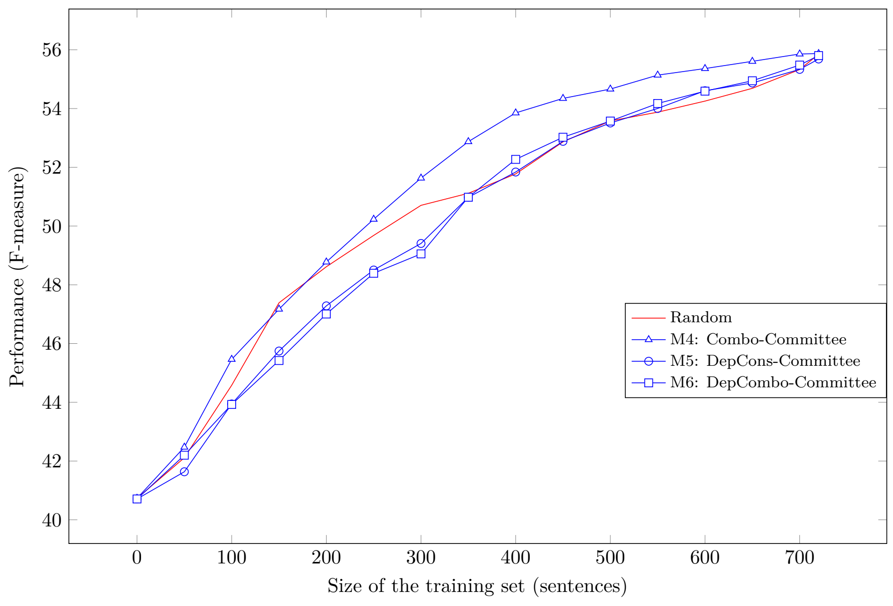


Figure 8.5: Learning curve of the learnt sentences in query-by-committee models and the baseline

Figure 8.5 represents the learning curves based on the average performance for each 5 iterations of the query-by-committee methods (Models 4 to 6), along with the learning curve of random sampling as the baseline. As can be seen in the curves, Model 4 beats the baseline, while Models 5 and 6, which use dependency relations for sampling rather than constituency trees, have a worse performance than the baseline. Model 4 has a slightly better performance compared with the baseline in early iterations, but from iteration 20, Model 4 significantly outperforms the baseline and the performance rises steadily until the end of the annotation process. Model 6, which has a slightly better performance than Model 5, loses against Model 5 until iteration 35, where it beats Model 5. This model has a relatively comparable performance with Model 5 to the end of the annotation process. In early iterations, Models 5 and 6 have a worse performance than the baseline, but they beat it in iteration 40. From these models, we can conclude that Model 4, which uses a committee of word- and class-based constituency models, can be selected as the best query-by-committee model to select informative samples.

Figure 8.6 represents the learning curves of Models 3 and 4, which are the best models of the entropy-based uncertainty sampling and query-by-committee sampling, in addition to random sampling and sentence length sampling methods as the baselines. As can be seen in the curves, Models 3 and 4 beat random sampling. Moreover, Model 3 outperforms Model 4 in the early iterations, but

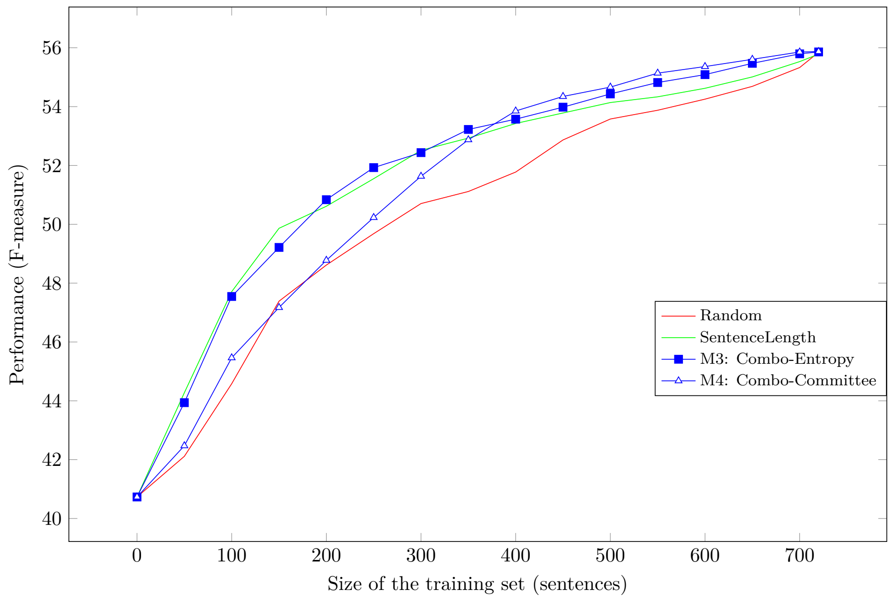


Figure 8.6: Learning curve of the learnt sentences in Combo-Entropy and Combo-Committee models and baselines

Model 4 beats it in iteration 37 which is almost half of the annotation process. From the results of the entropy-based uncertainty sampling method in Model 3 and the query-by-committee sampling method in Model 4, we can conclude that entropy-based uncertainty sampling with a meta-ranker is the best sampling method to be used for a real application. This method, however, has a shortcoming.

The general drawback of the entropy-based uncertainty sampling is that sentences which are relatively long and might contain complex syntactic constructions are selected as informative samples in early iterations. As a result, a huge amount of information is added to the training data in the iterative process. To study this point, we use sentence length as the second baseline and compare the performance of Models 3 and 4 against it. As can be seen in Figure 8.6, in early iterations, the sentence length model has a comparable performance with Model 3. This indicates that the performance of the entropy-based uncertainty sampling is comparable to a naive model that uses no intelligence to select the informative sentences but only uses the length of sentences.

In Figure 8.7, we display the learning curves of the baselines and Models 3 and 4 based on the learnt words rather than the sentences (see the x-axis). As shown in the figure, there is a competition between the models in early iterations. The performance of Model 3 is similar to random sampling, and the sentence length baseline has the worst performance. The low performance of the sentence

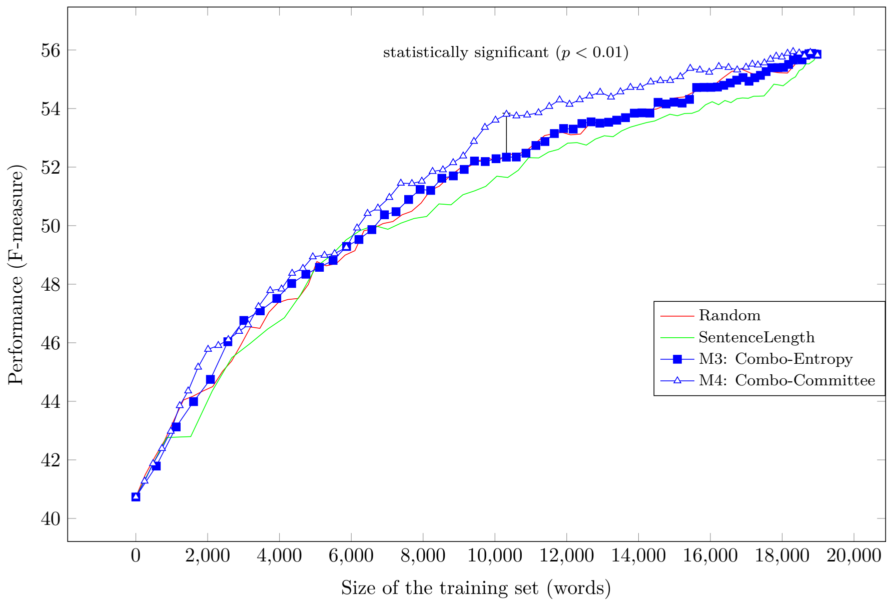


Figure 8.7: Learning curve of the learnt words in Combo-Entropy and Combo-Committee models and baselines

length baseline indicates that learning long sentences is not guaranteed to be informative and effective for the parser. Model 4 outperforms Model 3 and the baselines in iteration 23 by learning a minimum of 6,400 words. Learning this amount of words is almost a third of the words in the data pool. In iteration 36 where almost half of the words in the data pool are equally learnt by the parsers in the models, the difference between the performance of the parsers in Models 3 and 4 and the baselines in this iteration is statistically significant according to the 2-tailed t-test ($p < 0.01$). Consequently, Model 4 can be selected as the best model for treebanking with respect to the number of words learnt by the parser, since this model has obtained a higher performance when the parser learns a minimum number of words from the data pool.

So far, we studied the sampling methods based on the results of a parser; i. e. the performance of the parser is taken into consideration to select the informative samples and to compare different sampling models. We further study our proposed models from another perspective to find out what inside the selected sentences is to make them informative. To this end, we draw the active learning curves based on the similarity degree between the selected samples from the data pool and the training data (see the y-axis). This evaluation makes it possible to study the models from the linguistics point of view rather than their application.

In this study, we calculate the pairwise tree similarities between the parse

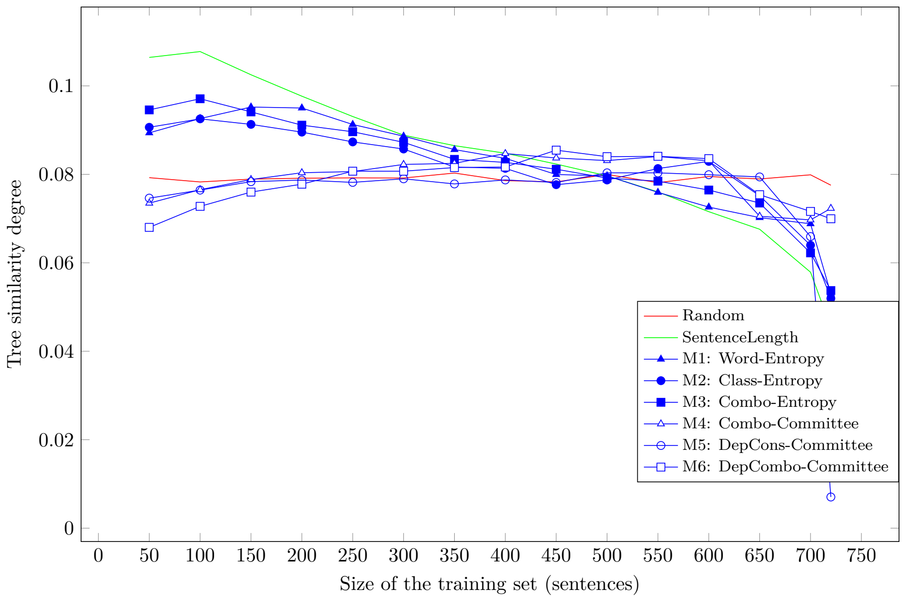


Figure 8.8: Learning curve of the learnt sentences based on tree similarity for entropy-based and committee models and baselines

trees of the selected samples in each iteration and all available parse trees in the training data. The average score of all pairwise similarities for each 5 iterations is represented in Figure 8.8. This figure is meaningful regarding the selected informative data in Models 1-6 and the baselines. As can be seen in the curves, random sampling has an almost flat curve, whereas after a small boost in tree similarity of the sentence length baseline, the similarity degree decreases constantly. The entropy-based uncertainty sampling models also have such a behavior. This result indicates that the selected informative samples in entropy-based uncertainty samplings are not linguistically informative, but they are informative for the parser. After examining the similarity curves of the query-by-committee models, we can conclude that in early iterations of these models, they try to find the most dissimilar samples, and the similarity degree consistently increases in the next iterations, and then they have flat curves, and finally the similarity degree decreases. We can conclude that the selected informative samples in early iterations of the query-by-committee models are linguistically more interesting than the entropy-based uncertainty sampling models. Among the query-by-committee models, Model 6 selectes the most dissimilar trees in early iterations. It should be recalled that among the query-by-committee models this model has the worst performance based on the learnt sentences.

Considering tree similarity as a sampling criterion to study and compare

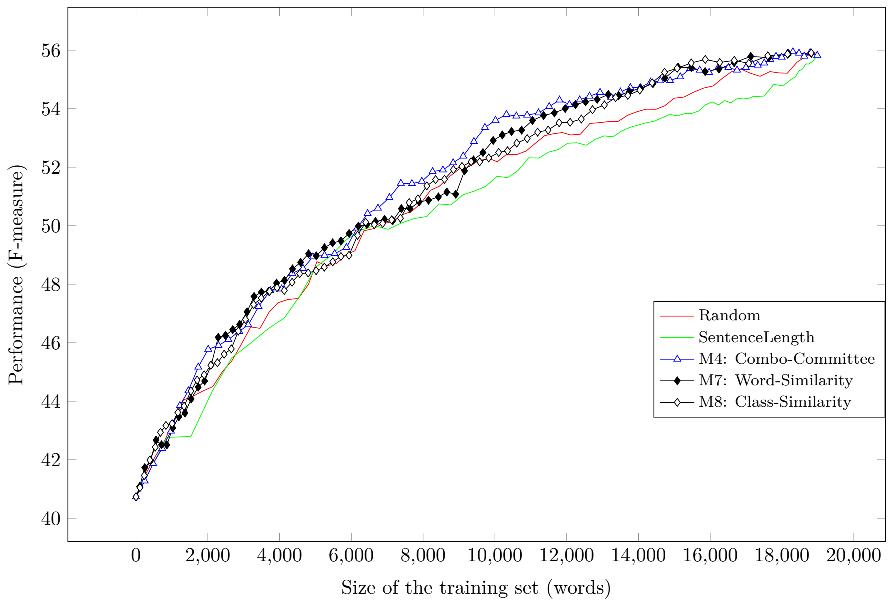


Figure 8.9: Learning curve of the learnt words in Combo-Committee and tree similarity models and baselines

different models motivated us to use this method for sampling. This sampling method helps to find the variability in the data pool that does not exist in the training data of the learner.

Figure 8.9 represents the learning curves of the tree similarity methods (Models 7 and 8) based on the learnt words. Additionally, the learning curves of the baselines and Model 4 are also demonstrated in this figure. As can be seen in the curves, the tree similarity models have relatively higher performance than the baselines. Among the tree similarity models, Model 7 has mostly better performance than Model 8, while Model 8 beats all models at a very late point when almost 75% of the words in the data pool are learnt. Moreover, in early iterations, Model 7 has a comparable performance with Model 4 to achieve a high performance with a minimum size of words to be learnt by the parser.

In Figure 8.10, we demonstrate the learning curves of models in Figure 8.8 in addition to the learning curves of the tree similarity models. As shown in the curves, the tree similarity models select the most dissimilar trees to the training data from the data pool in early iterations, and the similarity degree increases steadily as expected. In these models, the similarity degree is lower than random sampling and sentence length as the baselines which determines that the selected samples in Models 7 and 8 are linguistically informative either based on word forms, the corresponding POS tags, or syntactic constructions. Model 8 finds slightly more dissimilar trees in early iterations than Model 7,

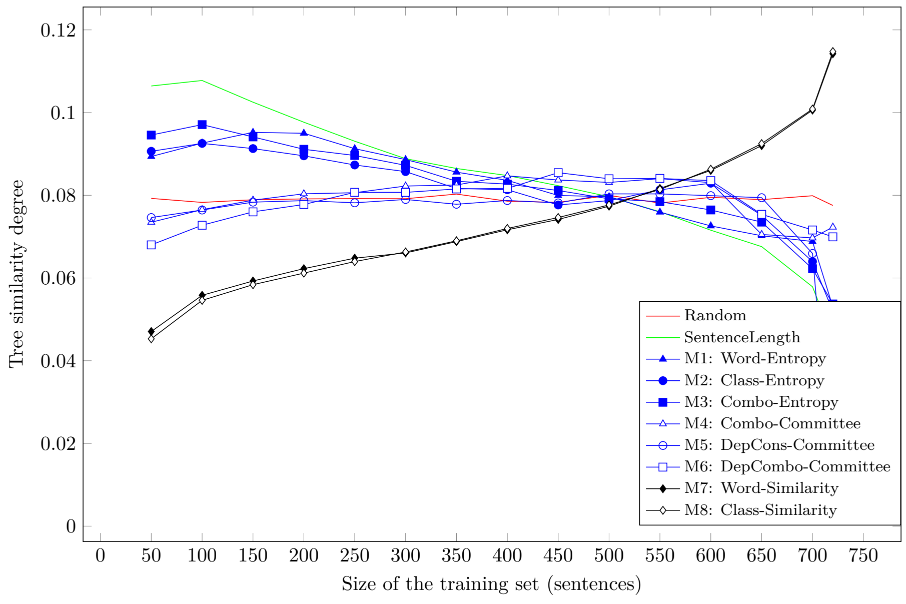


Figure 8.10: Learning curve of the learnt sentences based on tree similarity for all models and baselines

but the difference between the two models is not statistically significant. As demonstrated in the curves, the similarity degree of the two models increases steadily in the next iterations.

Based on the presented empirical results, we conclude that selecting a specific sampling method depends on the application. If we aim at increasing the performance of a parser, disregarding its shortcomings, entropy-based uncertainty sampling with a meta-ranker on the top of the word- and class-based models might be the best choice. If we want to increase the parsing performance with minimum number of words to be learnt by the parser, the committee of word- and class-based models for constituency parsing is a better alternative. However, if we want to select samples which are linguistically interesting, tree similarity models are the best options. The advantage of the query-by-committee and tree similarity models over the entropy-based uncertainty sampling models is that the query-by-committee and tree similarity models do not have the drawback of the entropy-based uncertainty sampling. The disadvantage of the query-by-committee models compared with the tree similarity models is that the query-by-committee models are more complex than the tree similarity models and they require at least two learners, whereas the tree similarity models require one learner and this model helps to develop a treebank with the most linguistically interesting variability on the data.

8.6 Summary

In this chapter, we employed two well-known sampling methods, namely uncertainty sampling and query-by-committee sampling, in active learning for Persian treebanking. In addition to random sampling, a naive sampling, which selects the longest sentences, was used as the baseline.

We used three entropy-based uncertainty sampling methods in which word- and class-based parsing and the combination of the two with a meta-ranker on the top were developed to select informative samples. The experimental results showed that the combo entropy-based uncertainty sampling, which benefits from both word- and class-based parsing, outperforms the individual word- and class-based uncertainty models. Although entropy-based sampling methods outperformed random sampling, comparing its result with sentence length determined that selecting relatively long sentences has been the main reason for achieving a good performance.

We also proposed query-by-committee models that exploited two parsing models for finding informative samples. These samples were the ones for which the parsers disagree. Our proposed models were constructed of word- and class-based parsing for either constituency or dependency parsers. In the models, the informative samples, disregarding their sentence length, were selected. Out of all of the models, the query-by-committee model which used word- and class-based constituency parsers achieved the best result.

We further studied the linguistic properties of the selected samples, and compared the models. We found out that in entropy-based uncertainty sampling models, the tree similarity degree of the selected samples relatively decreases, whereas in query-by-committee models it relatively increases. This means that query-by-committee models select samples which are linguistically more interesting. Moreover, when we used the tree similarity as a metric for sampling, we proposed word- and class-based models.

When the overall results were compared, we found out that the query-by-committee model composed of word- and class-based constituency parsers and the word-based tree similarity model have selected informative samples with the minimum number of the words learnt by the parser, and they have a slightly better performance than other models. Since the tree similarity sampling method used a simple criterion compared with the query-by-committee sampling and does not have the shortcoming of the entropy-based uncertainty sampling, this model can be selected as the best method to select linguistically interesting samples to enlarge our treebank.

Chapter 9

Conclusion

9.1 Summary

Recent studies have attempted to make machines understand natural languages. Learning the grammar of a language plays a major role in attaining this goal. The automatic parsing of a sentence is the preliminary step to achieve this. To train a statistical parser, a set of annotated data, called a treebank, is required. Not all languages are rich in terms of the availability of such a language resource. The motivation of the present research is proposing a method to develop a treebank for Persian from scratch according to the HPSG formalism. Since data annotation is a time-consuming task, we aim at using machine learning methods to reduce human intervention for data annotation. The developed annotated data can then be used as a data source to train statistical parsers. We described our ideas and the methodology in two parts.

In part I, we mainly concentrated on the theoretical level and the previous studies. Since Persian was our target language, we provided background for the general syntactic properties of this language and the problems one might face in corpus development and text processing of this language. Next, we reviewed the literature about grammar formalisms, including HPSG, along with treebanking, and parsing. To standardize our developed treebank, we explained an annotation scheme to be used throughout our treebanking.

In part II, we mainly focused on the computational perspective and applicability of the developed treebank. In this part, machine learning methods were exploited for developing and increasing the size of the treebank. To initialize treebanking, we employed a bootstrapping approach to select the most frequent grammar rules from the manually annotated data to be defined in the annotation tool for further usage. The proposed method significantly reduced human intervention during the data preparation process. The data set was then used

to train statistical constituency parsers. Availability of the information for the type of the dependency relations between the constituents based on the heads' argument structures made it possible to convert this treebank into its parallel dependency-based treebank. The converted data was used for training dependency parsers.
We further studied the effect of annotation granularity (fine- vs coarse-grained annotation) in three dimentions, namely lexical item, POS tag, and constituent label. We concluded that coarse-grained representation of the lexical items has a positive impact on parsing. We further concluded that using fine-grained constituent label reduces parsing performance due to increasing the complexity for determining the correct type of head-daughter dependencies. We also concluded that fine-grained morpho-syntactic information of POS tags plays an important role on parsing and losing this information causes reduction of performance.

The size of the developed treebank is very small. Due to the data sparsity problem of the developed treebank, a parser may not be able to create accurate grammar models. As a result, when the parser is used in a real application, it may not have a high performance. The data sparsity problem exists at two levels: at the lexical and at the syntactic construction levels. We proposed solutions to resolve the problem.

We proposed an unsupervised method to deal with the data sparsity problem at the lexical level. To this end, we used the Brown word clustering algorithm. In the class-based model, an unannotated corpus was used for clustering words into a pre-defined number of classes. We noted that one shortcoming of word clustering is that homographs are treated equally and since in Persian short vowels are not usually written, this problem is intensified. To deal with the problem, we attached the main POS tag of the words to the word forms to make a large portion of homographs distinct. Experimentally, we demonstrated the positive impact of using the class-based model for reducing the data sparsity problem at the lexical level. We further illustrated that using the extended version of clustering in the class-based parsing model improves the performance of the parser significantly.

To handle the data sparsity problem at the syntactic construction level, we proposed a supervised method. For this, we used active learning which is a promising machine learning method for selecting informative data from a corpus and asking a human to annotate the selected data. The consequence of this model is developing a data set which has a high impact on the learner with minimum increase in the data size and human effort for annotation. To reach the goal, we simulated the active learning process and employed several models to find the best one for further data annotation in order to enlarge the current Persian treebank.

We employed uncertainty sampling by computing the entropy of sentences for word- and class-based parsing models to select the informative samples. On top of these models, then, we constructed a meta-ranker to select informative samples based on the output of the word- and class-based models. The meta-ranker had a better learning process than the two individual models and the baseline. However, one drawback of the entropy-based uncertainty sampling was that the length of sentences has an adverse impact on selecting the samples. To resolve this problem, we employed several query-by-committee models for sampling. In these models, we used query-by-committee models composed of word- and class-based models. Dependency and constituency parsers were employed in the proposed models as well. The learning curves of the models demonstrated that the committee of word- and class-based constituency parsing performed the best. We therefore concluded that this model, which did not have the shortcoming of the entropy-based uncertainty sampling, could be used for further data annotation.
We further studied the uncertainty sampling from another perspective such that the similarity degree between the syntactic trees of the parser's output and the trees in the training data was computed. In this method, the most dissimilar trees in the data pool to the training data were selected as informative samples. This criterion was employed using the word- and class-based models. Comparing all proposed models, the word-based tree similarity model and the query-by-committee model composed of word- and class-based parsing demonstrated the best performance by learning the minimum lexical items.

9.2 Future Work

Possible continuations of the present research can go in several directions. One possibility is finding the minimum morpho-syntactic information that should exist in the POS tags of the words and studying its impact on the parsing process. As an example, which morpho-syntactic information, for instance between Ezāfe and plurality, has a greater impact on parsing? The consequence of this study is determining the most important morpho-syntactic information that is very informative for a parser and must be available in the POS tags of the words.

In our study, we employed a hard clustering approach for parsing, though our extended clustering model made this clustering approach soft to some extent. One possibility for further study will be using a soft clustering approach for parsing. In this approach, one word will be assigned to more than one cluster. The challenge of this approach can be to find the correct cluster that a word in the local context should map to.

We explained that active learning can be used for data annotation, and we found alternative models for this purpose. Another possibility to continue this research will be to use the proposed method for a real application to enlarge the current treebank. It is also possible to propose new sampling methods. One option can be to add more parsers to the committee members, therefore more learners are involved in making decisions. And yet another possibility will be to use the entropy of constituents instead of using the entropy of sentences to find the informative constituents as a criterion to find uncertain samples.

Stopping the active learning process at an ideal point before annotating all data in the data pool can also be considered as another possibility to continue the present research.

Appendix A

Persian Alphabet

Table A.1: List of the Persian alphabet (part 1: HAMZE-SIN)

Name	Letter Form				Phonetics	Letter Form				Unicode
	Non-joiner	Begin	Middle	End		Non-joiner	Begin	Middle	End	
HAMZE	ء ، أ ، ئ	ئـ	ـئـ	أ ، ـئ	ʔ	املاء ، رأس	رئیس	مسئول	هیأت، شیئ	u0621 u0623 u0626
ALEF with tilda	آ	–	–	ـآ	ā , ʔā	آب	–	–	مآخذ	u0622
ALEF	ا	–	–	ـا	ā , a , ʔa , e , ʔe , o , ʔo	دارو ، اکرم امروز، اروپا	–	–	شما	u0627
BE	ب	بـ	ـبـ	ـب	b	کتاب	برادر	سبک	سیب	u0628
PE	پ	پـ	ـپـ	ـپ	p	کاپ	پیر	سپاه	لامپ	u067E
TE	ت	تـ	ـتـ	ـت	t	فوت	توپ	متر	دست	u062A
SE	ث	ثـ	ـثـ	ـث	s	میراث	ثریا	مثبت	حدیث	u062B
JIM	ج	جـ	ـجـ	ـج	ʤ , ǰ	کاج	جار	لهجه	هویج	u062C
CHE	چ	چـ	ـچـ	ـچ	ʧ , č	قارچ	چاپ	کچل	گچ	u0686
HE	ح	حـ	ـحـ	ـح	h	لوح	حرم	صحرا	صلح	u062D
KHE	خ	خـ	ـخـ	ـخ	x	سوراخ	خاله	صخره	سیخ	u062E
DĀL	د	–	–	ـد	d	در	–	–	سبد	u062F
ZĀL	ذ	–	–	ـذ	z	لذت	–	–	ذلت	u0630
RE	ر	–	–	ـر	r	باران	–	–	سراسر	u0631
ZE	ز	–	–	ـز	z	روزنامه	–	–	سبز	u0632
ZHE	ژ	–	–	ـژ	ʒ , ž	ژاله	–	–	پژمرده	u0698
SIN	س	سـ	ـسـ	ـس	s	رأس	سبز	پسر	مگس	u0633

Table A.2: List of the Persian alphabet (part 2: SHIN-YE)

Name	Letter Form				Phonetics	Letter Form				Unicode
	Non-joiner	Begin	Middle	End		Non-joiner	Begin	Middle	End	
SHIN	ش	شـ	ـشـ	ـش	ʃ , š	موش	شیئ	کشک	کش	u0634
SĀD	ص	صـ	ـصـ	ـص	s	خاص	صبح	مصلح	خالص	u0635
ZĀD	ض	ضـ	ـضـ	ـض	z	مرض	ضایع	مضطرب	مریض	u0636
TĀ	ط	طـ	ـطـ	ـط	t	احتیاط	طناب	قطر	رابط	u0637
ZĀ	ظ	ظـ	ـظـ	ـظ	z	محفوظ	ظالم	منظم	حفظ	u0638
EYN	ع	عـ	ـعـ	ـع	ʔ	اشباع	علم	معلم	خلع	u0639
GHEYN	غ	غـ	ـغـ	ـغ	q	کلاغ	غایب	اشتغال	جیغ	u063A
FE	ف	فـ	ـفـ	ـف	f	معروف	فکر	کفر	کیف	u0641
GHĀF	ق	قـ	ـقـ	ـق	q	بوق	قاب	مقیم	مشق	u0642
KĀF	ک	کـ	ـکـ	ـک	k	ساک	کمد	عکس	سبک	u06A9
GĀF	گ	گـ	ـگـ	ـگ	g	بزرگ	گنجه	مگس	سگ	u06AF
LĀM	ل	لـ	ـلـ	ـل	l	کال	لذیذ	علم	حل	u0644
MIM	م	مـ	ـمـ	ـم	m	چرم	مادربزرگ	کمک	ظالم	u0645
NUN	ن	نـ	ـنـ	ـن	n	باران	نما	منظم	من	u0646
VĀV	و	–	–	ـو	v, o, – u, ow	ناو، دو روز، اوج	–	–	نانوا، نوک، خواهر بوسه، موز	u0648 u0648
HE	ه	هـ	ـهـ	ـه	h , –	ده	هراس	مهم	بِه، خانه	u0647
YE	ی	یـ	ـیـ	ـی	y , i	چای	مریم	سیاه	ماهی	u064A

Appendix B

EAGLES-based POS tags in ‘Peykare’

Table B.1: EAGLES-based POS tags for Persian in ‘Peykare’ (part-I)

	Recommended Attributes									Special Extensions			
Obligatory Attribute	Type	Number	Person	Mood	Tense	Copulation	Status	Person & Number	Degree	Generic Attribute	Language Specific Attributes		
										Semantic Features	Polarity	Enclitic	Fusion
Adverb (ADV)	general (GEN) intensifier (INTSF)								comparative (COM) simple (SIM)	(EXAM) locative (LOC) negative (NEG) order (ORD) interrogative word (QU) repetition (REPT) wish (WISH)		Ezāfe (EZ) pronominal (1,2,3,4,5,6)	conjunction (CONJ) post-position (PostP)
Adjective (ADJ)									comparative (COMP) simple (SIM) superlative (SUP)			Ezāfe (EZ) Ye (YE) Pronominal (1,2,3,4,5,6)	conjunction (CONJ) post-position (PostP)
Classifier (CL)		plural (PL) singular (SING)										Ezāfe (EZ) Ye (YE)	conjunction (CONJ)
Conjunction (CONJ)	conditioner (COND) correlative (CORR)											Ezāfe (EZ) pronominal (1,2,3,4,5,6)	adverb (ADV) conditioner (COND) determiner, demonstrative (DET,DEMO) (INT) noun (N) post-position (PostP) preposition, adverb (PREP,ADV) preposition, determiner (PREP,DET) preposition, post-position (PREP,PostP) preposition, pronoun (PREP,PRO) pronoun (PRO)
Determiner (DET)	demonstrative (DEMO) indefinite (INDF) interrogative words (QU)											Ezāfe (EZ) pronominal (1,2,3,4,5,6)	classifier (CL) determiner (DET)
Interjection (INT)													
Noun (N)	common (COM) proper (PR)	plural (PL) singular (SING)								acronym (ACR) day (DAY) infinitive (INFI) locative (LOC) month (MON) negation (NEG) season (SEAS) surname (SURN) time (TIME) vocative (VOC)		Ezāfe (EZ) Ye (YE) pronominal (1,2,3,4,5,6)	Conjunction (CONJ) Post-position (POSTP)

Table B.2: EAGLES-based POS tags for Persian in ‘Peykare’ (part-II)

Obligatory Attribute	Recommended Attributes									Special Extensions			
	Type	Number	Person	Mood	Tense	Copulation	Status	Person & Number	Degree	Generic Attribute	Language Specific Attributes		
										Semantic Features	Polarity	Enclitic	Fusion
Numeral (NUM)	cardinal (CAR) ordinal (ORD)	plural (PL) singular (SING)								adjective (AJC) nominative (NOMI)		Ezāfe (EZ) Ye (YE) pronominal (1,2,3,4,5,6)	classifier (CL) noun (N)
Preposition (PREP)												Ezāfe (EZ) pronominal (1,2,3,4,5,6)	determiner (DET) noun (N) pronoun (PRO)
Pronoun (PRO)	demonstrative (DEMO) indefinite (INDF) (PEFL) personal (PERS) reciprocal (REC)	plural (PL) singular (SING)	1 2 3									Ezāfe (EZ) Ye (YE) Pronominal (1,2,3,4,5,6)	conjunction (CONJ) post-position (PostP)
Post-position (PostP)													
Punct (PUNC)													
Residual (RES)	alphabet (ALPHAB) foreign word(FW) mathematical sign(MS) poem (POEM) phrasal sentence (PS) unclear (UNCL)									Arabic (ARAB) latin (LAT) Persian (PERSN)		Ezāfe (EZ)	
Verb (V)				(COPR) imperative (IMP) past participle (PASTP) subjunctive (SUB)	(EIMPERF) future (FUT) imperfect (IMPERF) past (PA) perfect (PERF) present (PRES)	adverbial complement (ADVC) adjectival complement (AJCC) nominative complement (NC) prepositional complement (PREPC) prepositional+nominative complement (PREPNC) pronominal complement (PROC)	auxiliary (AUX) non-inflectional (NIN)	1 2 3 4 5 6			negative (NEG) positive (POS)	pronominal (1,2,3,4,5,6)	conjunction (CONJ)

Appendix C

MulText-East Format of POS Tags in the Bijankhan Corpus

Table C.1: POS tag of Adjectives based on the MulText-East Framework

CATEGORY	POLARITY	TYPE	CLITIC
Adjective (A)	Positive (p)	Simple (s)	Ezafe (z)
	Negative (n)	Comparative (c)	Ya (y)
		Superlative (u)	

Table C.2: POS tag of Adverbs based on the MulText-East Framework

CATEGORY	TYPE	POLARITY	SUB-TYPE	SEMANTIC	FUSION	CLITIC
Adverb (D)	General (g)	Negative (n)	Simple (s)	Time (t)	Post-position (p)	Ezafe (z)
	Intensifier (i)	Positive (p)	Comparative (c)	EXAM (e)	Conjunction (j)	Ya (y)
				QU (q)		
				Location (k)		
				Order (o)		
				Repetition (r)		
				Wish (w)		

Table C.3: POS tag of Classifiers based on the MulText-East Framework

CATEGORY	NUMBER	CLITIC
Classifier (L)	Singular (s)	Ezafe (z)
	Plural (l)	Ya (y)

Table C.4: POS tag of Conjunctions based on the MulText-East Framework

CATEGORY	TYPE	CLITIC
Conjunction (J)	Condition (o)	Ezafe (z)
	Coordinator (r)	Ya (y)
	Restrictive (r)	
	Complementizer (m)	

Table C.5: POS tag of Interjections based on the MulText-East Framework

CATEGORY
Interjection (I)

Table C.6: POS tag of the Post-position based on the MulText-East Framework

CATEGORY
Post-position (P)

Table C.7: POS tag of Prepositions based on the MulText-East Framework

CATEGORY	CLITIC
Preposition (E)	Ezafe (z)
	Ya (y)

Table C.8: POS tag of Punctuation based on the MulText-East Framework

CATEGORY	TYPE
Punctuation (O)	End Position (e)
	Quotation Mark(q)
	Hash Sign (x)
	Dollar Sign (d)
	Ampersand (a)
	Bracket (b)
	Mathematical Signs (m)
	Comma (c)
	Hyphen (h)
	Etc (z)
	Underline (u)
	Colon (o)
	Question Mark (u)
	At Sign (t)
	Semi-colon (k)
	Sokun (s)

Table C.9: POS tag of Residuals based on the MulText-East Framework

CATEGORY	TYPE	NUMBER	SEMANTIC	CLITIC
Residual (R)	MS (m)	Singular (s)	Latin (t)	Ezafe (z)
	PS (p)	Plural (l)	Persian (f)	Ya (y)
	UNCL (u)		Arabic (b)	
	FW (w)			
	ALPHAB (a)			
	POEM (o)			
	MADJ (j)			
	MADV (v)			
	MN (n)			

Table C.10: POS tag of Clitics based on the MulText-East Framework

CATEGORY	TYPE	MINOR-POS	POLARITY	VERB-TYPE	HOST	NUMBER	PERSON	HOST-FINAL-LETTER
Clitic (C)	Enclitic (e)	Pronoun (z)	Positive (p)	Copula (k)	Adjective (a)	Singular (s)	First (o)	Consonant (c)
	Proclitic (p)	Verb (v)	Negative (n)		Noun (n)	Plural (l)	Second (t)	Vowel (v)
		Post-Position (p)			Preposition (e)		Third (h)	
		Conjunctor (j)			Adverb (d)			
		Preposition (e)			Pronoun (z)			
		Adverb (d)			Number (u)			
		Determiner (t)			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)			

Table C.11: POS tag of Determiners based on the MulText-East Framework

CATEGORY	TYPE	FUSION	Definiteness	SUB-TYPE	SEMANTIC	SUB-TYPE	NUMBER	CLITIC
Determiner (T)	Demonstrative (m)	Adverb (d)	Indefinite (f)	General (g)	Time (t)	Common (c)	Singular (s)	Ezafe (z)
	Indefinite (f)	Classifier (l)		Intensifier (i)	EXAM (e)	Proper (a)	Plural (l)	Ya (y)
	Quantifier (q)	Determiner (t)			QU (q)			
		Noun (n)			Location (k)			
					Order (o)			
					Repetition (r)			
					Wish (w)			

Table C.12: POS tag of Nouns based on the MulText-East Framework

CATEGORY	TYPE	NUMBER	POLARITY	SEMANTIC	ABBREVIATION	CLITIC
Noun (N)	Common (c)	Singular (s)	Negative (n)	Location (k)	Abbreviation (b)	Ezafe (z)
	Proper (a)	Plural (l)	Possitive (p)	Time (t)		Ya (y)
				Day (d)		
				Month (m)		
				Season (e)		
				Title (f)		
				Vocalic (v)		
				Direction (c)		

Table C.13: POS tag of Numbers based on the MulText-East Framework

CATEGORY	TYPE	LOCAL-FUNCTION	NUMBER	FUSION	CLITIC	SEMANTIC
Number (U)	Ordinal (o)	Adjective (a)	Singular (s)	Adjective (a)	Ezafe (z)	Time (t)
	Cardinal (r)	Noun (n)	Plural (l)	Classifier (l)	Ya (y)	

Table C.14: POS tag of Pronouns based on the MulText-East Framework

CATEGORY	TYPE	NUMBER	PERSON	LOCAL-FUNCTION	FUSION	CLITIC	PLURALITY
Pronoun (Z)	Demonstrative (m)	Singular (s)	First (o)	INO (b)	Post-position (p)	Ezafe (z)	Plural (l)
	Indefinite (f)	Plural (l)	Second (t)			Ya (y)	
	Reflexive (x)		Third (h)				
	Recursive (c)						
	Personal (r)						
	Interrogative (i)						

Table C.15: POS tag of Verbs based on the MulText-East Framework

CATEGORY	POLARITY	AUXILIARY/MAIN	TYPE	NUMBER	PERSON	TENSE	ASPECT	MOOD	CLITIC	IMPERSONAL-MODAL
Verb (V)	Negative (n)	Auxiliary (x)	Simple (s)	Singular (s)	First (o)	Present (s)	Perfect (f)	Subjunctive (u)	Ezafe (z)	Impersonal Modal (n)
	Possitive (p)	Main (y)	Light (b)	Plural (l)	Second (t)	Past (t)	Imperfect (i)	Imperative (m)	Ya (y)	
			Copula (k)		Third (h)	Future (u)	Cimperfect (q)	Copr (r)		
			Infinitive (i)					Past-Participle (p)		

Appendix D

Hierarchy of Dependency Relations

- *BOT* - bottom
  - *ROOT* - root
  - *DEP* - dependent
    - *PUNC* - punctuation
    - *ARG* - argument
      - *COMP* - complement
        - *ADJCOMP* - adjective complement
        - *ADVCOMP* - adverb complement
        - *AUX* - auxiliary
        - *COPCOMP* - copula complement
        - *COORCOMP* - coordination complement
        - *CCOMP* - clausal complement
        - *DETCOMP* - determiner complement
        - *INTCOMP* - interjection complement
        - *NN* - nominal complement
          - *POSS* - possession complement
          - *POSSESSIVE* - possessive complement
          - *REFLCOMP* - reflexive complement
        - *OBJ* - object
          - *DOBJ* - direct object
          - *IDOBJ* - indirect object
          - *POBJ* - preposition object
          - *XOBJ* - dropped object
        - *SUBJ* - subject

- - - *NSUBJ* - nominal subject
      - *RELSUBJ* - relativizer subject
      - *XSUBJ* - dropped subject
  - *MOD* - modifier
    - *ADJMOD* - adjective modifier
    - *CLMOD* - clausal modifier
    - *DETMOD* - determiner modifier
    - *MWE* - multi-word expression modifier
    - *INTMOD* - interjection modifier
    - *NMOD* - noun modifier
      - *NUM* - numeric modifier
      - *APPOS* - appositional modifier
      - *REFLMOD* - reflexive modifier
    - *NUMBER* - element of compound number numeric
    - *PREPMOD* - preposition modifier
    - *PURPCL* - purpose clause modifier
    - *RCMOD* - relative clause modifier
    - *RES* - residual modifier
    - *VMOD* - verb modifier
- *CC* - coordination
  - *ADJCOOR* - adjective coordination
  - *ADVCOOR* - adverb coordination
  - *CLCOOR* - clause coordination
  - *DETCOOR* - determiner coordination
  - *NOUNCOOR* - noun coordination
  - *NUMCOOR* - number coordination
  - *PREPCOOR* - preposition coordination
  - *RESCOOR* - residual coordination
  - *VERVCOOR* - verb coordination
- *MARKER* - marker
  - *ACC* - accusative marker
  - *COMPM* - complementizer marker
  - *MARK* - clausal marker

.

Bibliography

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

AbolhasaniChime, Zahra and Masood Ghayoomi. enzemām: farāyande vāžesāziye harfe ezāfe va payāmadhāye kārbordiye ān dar zabānšenāsiye rāyāneyi [“Incorporation: Word production of Persian prepositions and its application in computational linguistics”]. In *Proceedings of the 2nd Workshop on the Persian Language and Computer*, pages 16–24, 2006.

AbolhassaniChime, Zahra. An account for compound preposition in Farsi. In *Proceedings of the International Conference on Computational Linguistics and Association for Computational Linguistics*, pages 113–119, 2006.

Adjukiewicz, Kazimierz. Die syntaktische Konnexität. *Studia Philosophica*, 1: 1–27, 1935. “Syntactic Connexion” by H. Weber in McCall, S. (Ed.) *Polish Logic*, pp. 207–231, Oxford University Press, Oxford, 1967.

Aghaei, Behrad. *Clausal Complementation in Modern Persian.* PhD thesis, University of Texas at Austin, 2006.

Aho, Alfred V., Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools.* Pearson Education, Inc., Boston, MA, USA, 2 edition, 2007.

Aldezabal, Izaskun, Koldo Gojenola, and Kepa Sarasola. A bootstrapping approach to parser development. In *International Workshop on Parsing Technologies*, pages 17–28, 2000.

AleAhmad, Abolfazl, Hadi Amiri, Ehsan Darrudi, Masoud Rahgozar, and Farhad Oroumchian. Hamshahri: A standard Persian text collection. *Knowledge-Based Systems*, 22(5):382–387, 2009.

Allen, James. *Natural Language Understanding.* The Benjamin/Cummins Publishing Company, Redwood City, California, 2 edition, 1995.

Amtrup, Jan W., Hamid Mansouri-Rad, Karine Megerdoomian, and Rémi Zajac. Persian-English machine translation: An overview of the Shiraz project. Memoranda in Computer and Cognitive Science MCCS-00-319, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 2000.

Aono, Masaki and Hironori Doi. A method for query expansion using a hierarchy of clusters. In *Proceedings of the Asian Information Retrieval Symposium*, pages 479–484, 2005.

Arabsorkhi, Mohsen, Hesham Faili, and Mansoor Zolghadri Jahroumi. Using genetic algorithm for Persian grammar induction. In *Proceedings of the 2009 IEEE International Conference on Natural Language Processing and Knowledge Engineering*, pages 146–151, September 24–27 2009.

Assi, Mostafa and Mohammad HajiAbdolhosseini. Grammatical tagging of a Persian corpus. *International Journal of Corpus Linguistics*, 5(1):69–82, 2000.

Assi, SeyyedMostafa. Farsi linguistic database (FLDB). *International Journal of Lexicography*, 10(3):5, 1997.

Assi, SeyyedMostafa. Persian language and IT. In *Proceedings of the 2nd Workshop on Information Technology and Its Disciplines*, pages 85–94, Kish Island, Iran, 2004.

Assi, SeyyedMostafa. PLDB: Persian Linguistics DataBase. *Pažuhešgarān [Researchers]*, 2005.

Assi, SeyyedMostafa. pāygāhe dādeye zabāne fārsi dar internet [‘The Persian linguistics data base on Internet’]. Technical report, Institute for Humanities and Cultural Studies, Tehran, Iran, 2006. URL `http://pldb.ihcs.ac.ir/Files/PLDB-REPORT85.pdf`.

Avgustinova, Tania and Yi Zhang. Conversion of a Russian dependency treebank into HPSG derivations. In *The 9th International Workshop on Treebanks and Linguistic Theories*, pages 7–18, 2010.

Ayat, Maryam. *yek gerāmere mohāsebāti barāye zabāne fārsi [“A Computational Grammar for the Persian Language”]*. Master’s thesis, Department of Computer Engineering and IT, Amirkabir University, Tehran, Iran, 2002.

Bagherbeygi, Somayeh and Mehrnoush Shamsfard. Corpus based semi-automatic extraction of Persian compound verbs and their relations. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 8ht International Conference on Language Resources and*

Evaluation, Istanbul, Turkey, May 23–25 2012. European Language Resources Association.

Bahrani, Mohammad, Hossein Sameti, and Mehdi HafeziManshadi. A computational grammar for Persian based on GPSG. *Language Resources and Evaluation*, 45(4):387–408, 2011.

Baldridge, Jason and Miles Osborne. Active learning for HPSG parse selection. In *Proceedings of the 7th Conference on Natural Language Learning at Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics*, pages 17–24, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Baldridge, Jason and Miles Osborne. Active learning and the total cost of annotation. In Lin, Dekang and Dekai Wu, editors, *Proceedings of Empirical Methods in Natural Language Processing*, pages 9–16, Barcelona, Spain, 2004.

Baldridge, Jason and Miles Osborne. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*, 14(2):191–222, 2008.

Baram, Yoram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithm. *Journal of Machine Learning Research*, pages 255–291, 2004.

Bateni, MohammadReza. *tousife sāxtemāne dasturiye zabāne fārsi ["A Description of the Grammatical Structure of the Persian Language"]*. Amikabir, Tehran, Iran, 1969.

Bezdek, James C., Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191–203, 1984.

Bijankhan, Mahmood. naqše peykarehāye zabāni dar neveštane dasture zabān: mo‘arrefiye yek narmafzāre rāyāneyi ["The role of corpora in writing a grammar: Introducing a software"]. *Journal of Linguistics*, 19(2):48–67, 2004.

Bijankhan, Mahmood. motāle?e va tahqiq jahate tadvine pažuhešnāmeye amaliyātiye dādegān: piyādesāziye estāndārde igolz dar peykareye matniye zabāne fārsiye moaser ["The study and research for a database development: Applying the EAGLES guidelines on the contemporary Persian text corpus"]. Corpus annotation technical report, Supreme Council of Information and Communication Technology and the University of Tehran–Iran, 2007. URL `http://www.scict.ir/portal/File/ShowFile.aspx?ID=c93e415f-f977-46bd-8545-911867752e2b`.

Bijankhan, Mahmood, Javad Sheykhzadegan, Ali Darzi, Hosein RaziZadeh, MohammadEsma'il Ghasedi, Javad Bagheri, Vahid Sadeghi, Zahra Mahmoudzadeh, Maryam DanayTousi, and Arezoo Moazzemi. peykareye matniye zabāne fārsi ["The Persian text corpus"]. In *Proceedings of the 1st Workshop on the Persian Language and Computer*, pages 143–144, University of Tehran, Iran, 2004.

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

Bird, Steven and Edward Loper. NLTK: The natural language toolkit. In *The Companion Volume to the Proceedings of the Association for Computational Linguistics*, pages 214–217, Barcelona, Spain, 2004. Association for Computational Linguistics.

Bird, Steven, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., Sebastopol, CA, USA, 2009.

Black, Ezra W., Steven Abney, Daniel P. Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. Procedure for quantitatively comparing the syntactic coverage of English grammars. In Black, E., editor, *Proceedings of the workshop on Speech and Natural Language*, pages 306–311, 1991.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Vidová-Hladká. *The Prague Dependency Treebank: A Three-Level Annotation Scenario*, chapter 7, pages 103–127. Kluwer Academic Publishers, The Netherlands, 2003.

Bohnet, Bernd. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*, pages 67–72, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Bonami, Olivier and Pollet Samvelian. Inflectional periphrasis in Persian. In *Proceedings of the 16th International Conference on Head-driven Phrase Structure Grammar*, pages 26–46, Stanford, California, 2009. CSLI Publications.

Bos, Johan, Cristina Bosco, and Alessandro Mazzei. Converting a dependency-based treebank to a categorial grammar treebank for Italian. In Passarotti, M., Adam Przepiórkowski, S. Raynaud, and Frank Eyndevan , editors, *Proceedings of the 8th Workshop on Treebanks and Linguistic Theories*, pages 27–38, 2009.

Boullier, Pierre and Benoît Sagot. Efficient and robust LFG parsing: SXLFG. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 1–10, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the 1st Workshops on Treebanks and Linguistic Theories*, pages 24–41, 2002.

Brants, Thorsten. The NeGra export format for annotated corpora. CLAUS Report 98, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany, 1997.

Brants, Thorsten. Cascaded Markov models. In *Proceedings of the 9th Conference on European Chapter of the Association for Computational Linguistics*, pages 118–125, 1999a.

Brants, Thorsten. *Tagging and Parsing with Cascaded Markov Models: Automation of Corpus Annotation.* PhD thesis, German Research Center for Artificial Intelligence and Saarland University, Saarbrücken, Germany, 1999b.

Brants, Thorsten. TnT - A statistical part-of-speech tagger. In *Proceedings of the Association for Neuro-Linguistic Programming and the North American Chapter of the Association for Computational Linguistics*, pages 224–231, 2000.

Bresnan, Joan and Ronald M. Kaplan. *The Mental representation of grammatical relations.* MIT press series on cognitive theory and mental representation. The MIT Press, Cambridge (Massachusett), London, 1982.

Brown, Peter F., Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

Brunstein, Ada. Annotation guidelines for answer types: BBN technologies, 2002. URL `http://catalog.ldc.upenn.edu/docs/LDC2005T33/BBN-Types-Subtypes.html`.

Bulyko, Ivan and Mari Ostendorf. A bootstrapping approach to automating prosodic annotation for limited-domain synthesis, 2002.

Burke, Michael. *Automatic Treebank Annotation for the Acquisition of LFG Resources*. PhD thesis, Dublin City University, 2006.

Busser, Bertjan and Roser Morante. Designing an active learning based system for corpus annotation. In *Revista de Procesamiento del Lenguaje Natural*, number 35, pages 375–381, 2005.

Cahill, Aoife, Mairead Mccarthy, Josef Van Genabith, and Andy Way. Automatic annotation of the Penn treebank with LFG f-structure information. In *LREC Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, pages 8–15, 2002.

Callmeier, Ulrich. PET – A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107, 2000.

Candito, Marie and Benoît Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141, Paris, France, 2009.

Candito, Marie and Djame Seddah. Parsing word clusters. In *Proceedings of the Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics, First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–84, Los Angeles, California, 2010.

Candito, Marie, Benoît Crabbé, and Pascal Denis. Statistical French dependency parsing: Treebank conversion and first results. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 1840–1847, La Valletta, Malta, 2010. European Language Resources Association.

Candito, Marie, Enrique Henestroza Anguiano, and Djame Seddah. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of the 12th International Conference on Parsing Technology*, pages 37–42, Dublin City University, 2011.

Carroll, John, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: A survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, 1998.

Carter, David M. The TreeBanker: A tool for supervised training of parsed corpora. In *Proceedings of the Workshop On Computational Environments For Grammar Development And Linguistic Engineering*, pages 9–15, 1997.

Çakici, Ruken. Automatic induction of a CCG grammar for Turkish. In *Proceedings of the ACL Student Research Workshop*, pages 73–78, 2005.

Chanev, Atanas, Kiril Simov, Petya Osenova, and Svetoslav Marinov. Dependency conversion and parsing of the BulTreeBank. In *Proceedings of the LREC workshop Merging and Layering Linguistic Information*, pages 16–23, 2006.

Charniak, Eugene. Tree-bank grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1031–1036, 1996.

Charniak, Eugene. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, pages 132–139, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Chen, Jinying, Andrew Schein, Lyle Ungar, and Martha Palmer. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 120–127, Stroudsburg, PA, USA, 2006a. Association for Computational Linguistics.

Chen, John, Srinivas Bangalore, and Vijay K. Shanker. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299, 2006b.

Chen, Wenliang, Xingzhi Chang, Huizhen Wang, Jingbo Zhu, and Tianshun Yao. Automatic word clustering for text categorization using global information. In *Proceedings of the Asian Information Retrieval Symposium*, volume 3411 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2004.

Chomsky, Noam. *Syntactic structures.* Mouton (Hague), Berlin, 1957.

Chomsky, Noam. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

Chomsky, Noam. Minimalist inquiries: The framework. In Martin, R., D. Michaels, and J. Uriagereka, editors, *Step by step: Essays in honor of Howard Lasnik.* The MIT Press, Cambridge, MA, USA, 2000.

Chomsky, Noam. Derivation by phase. In Kenstowicz, Michael, editor, *Ken Hale: A Life in Language.* The MIT Press, Cambridge, MA, USA, 2001.

Church, Kenneth Ward. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143, Stroudsburg, PA, USA, 1988.

Čmejrek, Martin, Jan Cuřín, Jiří Havelka, Jan Hajič, and Vladislav Kuboň. Prague Czech-English dependency treebank: Syntactically annotated resources for machine translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1597–1600. European Language Resources Association, 2004.

Cocke, John. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, 1969.

Collins, Michael. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

Collins, Michael and Nigel Duffy. Convolution kernels for natural language. In *Proceedings of the 14th Conference on Advances in Neural Information Processing Systems*, pages 625–632, Cambridge, MA, USA, 2001. The MIT Press.

Collins, Michael and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the Association for Computational Linguistics*, pages 263–270, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Copestake, Ann. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, 2002.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 4 (3):281–332, 2006.

Cramer, Bart and Yi Zhang. Constraining robust constructions for broad-coverage parsing with precision grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 223–231, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Craswell, Nick and David Hawking. Overview of the TREC 2004 web track. In *Proceedings of the Thirteenth Text REtrieval Conference*. National Institute of Standards and Technology (NIST), 2004.

DabirMoghadam, Mohammad. majhul dar zabāne fārsi [“Passive in the Persian language”]. *maǰalleye zabānšenāsi [Journal of Linguistics]*, 2(1):31–64, 1986.

DabirMoghaddam, Mohammad. Compound verbs in Persian. *Studies in the Linguistic Sciences*, 27(2):25–59, 1997.

Darrudi, Ehsan, MohammadReza Hejazi, and Farhad Oroumchian. Assessment of a modern Farsi corpus. In *Proceedings of the 2nd Workshop on Information Technology and Its Disciplines*, pages 73–77, Kish Island, Iran, 2004.

Darzi, Ali. *Word Order, NP Movement, and Opacity Conditions in Persian.* PhD thesis, University of Illinois at Urbana-Champaign, 1996.

Marneffe, Marie-Catherinede and Christopher D. Manning. The Stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Dehdari, Jon and Deryle Lonsdale. A link grammar parser for Persian. In Karimi, Simin, Vida Samiian, and Don Stilo, editors, *Aspects of Iranian Linguistics*, volume 1. Cambridge Scholars Press, 2008.

Dhillon, Inderjit S., Subramanyam Mallela, and Raul Kumar. Enhanced word clustering for hierarchical text classification. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 191–200, 2002.

Diab, Mona T. An unsupervised approach for bootstrapping Arabic sense tagging. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, pages 43–50, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Dipper, Stefanie. Grammar-based corpus annotation. In Abeillé, Anne, Thorsten Brants, and Hans Uszkoreit, editors, *Proceedings of the Workshop on Linguistically Interpreted Corpora*, pages 56–64, 2000.

Dridan, Rebecca and Timothy Baldwin. Unsupervised parse selection for HPSG. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 694–704, 2010.

Earley, Jay. *An Efficient Context Free Parsing Algorithm.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, USA, 1968.

Edwards, Anthony William Fairbank. *Likelihood.* Johns Hopkins University Press, Baltimore, USA, 2 edition, 1992.

Eisner, Jason M. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

FahimNiya, Farzin. *moškelāte āmuzeš va yādgiriye xatte fārsi dar dānešāmuzāne sāle dovvome ebtedāyi [“Problems in Teaching and Learning Persian Script for the Second-grade Primary Students”].* Master’s thesis, Institute for Humanities and Cultural Studies, Iran, 2002.

Faili, Hesham. *estentāje esteqrāyiye geraāmere ehtemālātiye yek zabāne tabi‘i be raveše bimorabbi [“Unsupervised Grammar Induction for a Natural Language”].* PhD thesis, Computer Engineering Department, Sharif University of Technology, Tehran, Iran, 2006.

Ferret, Olivier and Brigitte Grau. A bootstrapping approach for robust topic analysis. *Natural Language Engineering*, 8(3):209–233, 2002.

Flickinger, Dan. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000.

Flickinger, Daniel, Yi Zhang, and Valia Kordoni. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–96, Lisbon, Portugal, 2012. Edições Colibri.

Forney, G. David. The Viterbi algorithm. In *Proceedings of the IEEE*, volume 61, pages 268–278, 1973.

Foth, Kilian A. and Wolfgang Menzel. Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and the International Conference of the Association for Computational Linguistics*, pages 321–328, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Freund, Yoav, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. In *Machine Learning*, volume 28, pages 133–168, Hingham, MA, USA, 1997. Kluwer Academic Publishers.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar.* Harvard University Press, 1985.

Gerdes, Kim and Pollet Samvelian. A statistical approach to Persian light verb constructions. In *Proceedings of the 27th International Conference on Lexis and Grammar*, 2008.

Ghayoomi, Masood. *pišbiniye vāže dar pardāzeše rāyāneyiye zabāne fārsi [“Word Prediction in Computational Processing of the Persian Language”].* Master’s thesis, Islamic Azad University, Tehran Central Branch, Iran, 2004.

Ghayoomi, Masood. Using variance as a stopping criterion for active learning of frame assignment. In *Proceedings of the Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics, Workshop on Active Learning for Natural Language Processing*, pages 1–9, Los Angeles, USA, 2010.

Ghayoomi, Masood. Bootstrapping the development of an HPSG-based treebank for Persian. *Linguistic Issues in Language Technology*, 7(1), 2012a.

Ghayoomi, Masood. From grammar rule extraction to treebanking: A bootstrapping approach. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 1912–1919, Istanbul, Turkey, 2012b.

Ghayoomi, Masood. Word clustering for Persian statistical parsing. In Isahara, Hitoshi and Kyoko Kanzaki, editors, *Advances in Natural Language Processing*, volume 7614 of *Lecture Notes in Computer Science: JapTAL ’12: Proceedings of the 8th International Conference on Advances in Natural Language Processing*, pages 126–137. Springer Berlin Heidelberg, 2012c.

Ghayoomi, Masood. mo’arrefiye dādegāne deraxti va tajziyegare xodkāre fārsi [“Introducing a treebank and a statistical parser for Persian”]. In *Proceedings of the 8th Conference of Iranian Linguistics*, volume 2, pages 666–679, 2013.

Ghayoomi, Masood and Bruno Guillaume. Interaction grammar for the Persian language: Noun and adjectival phrases. In *Proceedings of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing, 7th Workshop on Asian Language Resources*, pages 107–114, Suntec, Singapore, 2009. Association for Computational Linguistics.

Ghayoomi, Masood and Jonas Kuhn. Sampling methods in active learning for treebanking. In *Proceedings of the 12th International Workshop on Treebanks and Linguistic Theories*, pages 49–60, Sofia, Bulgaria, 2013.

Ghayoomi, Masood and Jonas Kuhn. Converting an HPSG-based treebank into its parallel dependency-based treebank. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 802–809, Reykjavik, Iceland, 2014.

Ghayoomi, Masood and Omid Moradiannasab. The effect of treebank annotation granularity on parsing: A comparative study. In *Proceedings of the 11th*

International Workshop on Treebanks and Linguistic Theories, pages 109–114, Lisbon, Portugal, 2012. Edições Colibri.

Ghayoomi, Masood and Stefan Müller. Multi-token units and multi-unit tokens in developing an HPSG-based treebank for Persian. In *4th International Conference on Iranian Linguistics*, page 29, 2011.

Ghayoomi, Masood, Saeedeh Momtazi, and Mahmood Bijankhan. A study of corpus development for Persian. *International Journal on Asian Language Processing*, 20(1):17–33, 2010.

Ghayoomi, Masood, Kiril Simov, and Petya Osenova. Constituency parsing of Bulgarian: Word- vs class-based parsing. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 4056–4060, Reykjavik, Iceland, 2014.

Gholamalizadeh, Khosro. *sāxte zabāne fārsi ["The Structure of the Persian Language"].* Ehyaye Ketāb, Tehran, Iran, 1999.

Ghomeshi, Jila. *Projection and Inflection: A Study of Persian Phrase Structure.* PhD thesis, University of Toronto, 1996.

Goldberg, Adele E. Words by default: The Persian complex predicate construction. In Francis, Elaine and Laura Michaelis, editors, *Linguistic Mismatches*, pages 117–146. CSLI Publications, 2003.

Goodman, Joshua. Parsing algorithms and metrics. In *Proceedings of the Association for Computational Linguistics*, pages 177–183, 1996.

Guo, Yuqing, Josef Genabithvan , and Haifeng Wang. Treebank-based acquisition of LFG resources for Chinese. In *Proceedings of the 12th International Lexical Functional Grammar Conference*, Stanford, California, 2007. CSLI Publications.

Hajič, Jan. Building a syntactically annotated corpus: The Prague dependency treebank. In Hajičová, E., editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 106–132. Karolinum, Prague, Czech Republic, 1998.

Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. Announcing Prague Czech-English dependency treebank 2.0. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3153–3160, Istanbul, Turkey, 2012. European Language Resources Association.

Hall, Johan, Joakim Nivre, and Jens Nilsson. A hybrid constituency-dependency parser for Swedish. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 284–287, 2007.

Han, Chung-hye, Na-Rare Han, Eon-Suk Ko, and Martha Palmer. Development and evaluation of a Korean treebank and its application to NLP. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1635–1642. European Language Resources Association, 2002.

Hays, David G. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525, 1964.

Heid, Ulrich. A linguistic bootstrapping approach to the extraction of term candidates from German text. *Terminology*, 5(2):161–181, 2000.

Hindle, Donald. User manual for Fidditch, a deterministic parser. Technical Memorandum 7590-142, Naval Research Laboratory, 1983.

Hockenamier, Julia and Mark Steedman. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 2007.

Hockenmaier, Julia. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK, 2003.

Hodge, Victoria J. and Jim Austin. Hierarchical word clustering - automatic thesaurus generation. *Neurocomputing*, 48:819–846, 2002.

Huang, Chu-Ren, Feng-Yi Chen, Keh-Jiann Chen, Zhao Gaosming , and Kuang-Yu Che. Sinica treebank: Design criteria, annotation guidelines, and on-line interface. In *Proceedings of 2nd Chinese Language Processing Workshop, Association for Computational Linguistics*, 2000.

Huang, Liang and David Chiang. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*, pages 53–64, Vancouver, BC, 2005.

Hughes, Baden, James Haggerty, Saritha Manickam, Joel Nothman, and James R. Curran. A distributed architecture for interactive parse annotation. In *In Proceedings of the Australasian Language Technology Workshop*, pages 207–214, 2005.

Hwa, Rebecca. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint special interest group for linguistic data and corpus-based approaches to NLP on Empirical Methods on Natural Language Processing and very large corpora held in conjunction with the Association for Computational Linguistics*, pages 45–52, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Hwa, Rebecca. On minimizing training corpus for parser acquisition. In Daelemans, Walter and Rémi Zajac, editors, *Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL-2001)*, pages 84–89. Toulouse, France, 2001.

Hwa, Rebecca. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276, 2004.

Ingerman, Peter Zilahy. *A Syntax-oriented Translator.* Academic Press, New York and Lonodn, 1966.

Iosif, Elias, Georgios Petasis, and Vangelis Karkaletsis. *Ontology-based Information Extraction under a Bootstrapping Approach*, chapter 1, pages 1–21. IGI Global, Hershey, PA, USA, April 2012.

Joshi, Aravind K. *Tree Adjoining Grammars: How Much Context Sensitivity is Required to provide Reasonable Structural Descriptions?*, chapter 6, pages 206–250. Cambridge University Press, 1985.

Jurafsky, Daniel and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, Upper Saddle River, New Jersey, 2000.

Kahnemuyipour, Arsalan. Persian Ezafeh construction revisited: Evidence for modifier phrase. In Jensen, J.T. and G. Herkvan , editors, *Proceedings of the 2000 Annual Conference of the Canadian Linguistic Association*, pages 173–185. Cahiers Linguistique d'Ottawa, 2000.

Kahnemuyipour, Arsalan. Persian Ezafeh construction: Case, agreement or something else. In *Proceedings of the 2nd Workshop on Persian Language and Computer*, University of Tehran, Iran, 2006.

Kalbassi, Iran. *sāxte ešteqāqiye vāže dar fārsiye emruz ["The Derivational Structure of Word in Modern Persian"].* Institute for Humanities and Cultural Studies, Tehran, Iran, 2001.

Kaplan, Ronald M. and Joan Bresnan. *Lexical-Functional Grammar: A Formal System for Grammatical Representation*, chapter 2, pages 29–130. Center for the Study of Language and Information Publication Lecture Notes. CSLI Publications, Stanford, California, 1995. URL `http://www2.parc.com/isl/groups/nltt/papers/kb82-95.pdf`.

Karimi, Simin. Persian compound verbs: Compositional or idiomatic. *Lexicology*, 3(2):273–318, 1997.

Karimi, Simin. Specificity effect: Evidence from Persian. *The Linguistic Review*, 16(2):125–141, 1999.

Karimi, Simin. *On Object Positions, Specificity, and Scrambling in Persian*, chapter 5, pages 91–125. Blackwell Publishing Ltd, Oxford, UK, 2003.

Karimi, Simin. *A Minimalist Approach to Scrambling: Evidence from Persian.* Mouton de Gruyter, 2005.

KarimiDoostan, GholamHosein. *Light Verb Constructions in Persian and Kurdish.* PhD thesis, University of Essex, 1997.

Kasami, Tadao. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA, 1965.

Keller, Frank. Towards an account of extraposition in HPSG. In *Proceedings of the 7th Conference on European Chapter of the Association for Computational Linguistics*, pages 301–306, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

Kiani, Soheila, Tara Akhavan, and Mehrnoush Shamsfard. Developing a Persian chunker using a hybrid approach. In *International Multi-conference on Computer Science and Information Technology*, pages 227–234, October 12-14 2009.

King, Paul. *A Logical Formalism for Head-driven Phrase Structure Grammar.* PhD thesis, Department of Mathematics, University of Manchester, 1989.

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

Klein, Dan and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of the Association for Computational Linguistics (ACL'02)*, pages 128–135, 2002.

Klein, Dan and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

Kneser, Reinhard and Jochen Peters. Semantic clustering for adaptive language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE Computer Society, 1997.

Koller, Alexander and Stefan Thater. Efficient solving and exploration of scope ambiguities. In *Proceedings of the Association for Computational Linguistics Interactive Poster and Demonstration Sessions*, pages 9–12. Association for Computational Linguistics, Ann Arbor, 2005.

König, Esther and Wolfgang Lezius. The TIGER language - A description language for syntax graphs - part 1: User's guidelines. Technical report, IMS, University of Stuttgart, 2002a.

König, Esther and Wolfgang Lezius. The TIGER language - A description language for syntax graphs - part 2: Formal definition. Technical report, IMS, University of Stuttgart, 2002b.

Koo, Terry, Xaviar Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of the Association for Computational Linguistics (ACL-08)*, pages 595–603, Colimbus, USA, 2008.

Kübler, Sandra, Wolfgang Maier, Ines Rehbein, and Yannick Versley. How to compare treebanks. In Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the 6th International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), May 28–30 2008.

Lambek, Joachim. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.

Lari, Karim and Steve J. Young. The estimation of stochastic context-free grammars using the inside–outside algorithm. *Computer Speech and Language*, 4: 35–56, 1990.

Laws, Florian and Hinrich Schütze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 465–472, Manchester, 2008.

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

Levenshtein, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966. URL `http://profs.sci.univr.it/~liptak/ALBioinfo/files/levenshtein66.pdf`.

Lewis, David D. and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

Lezius, Wolfgang. *Ein Werkzeug zur Suche für Syntaktisch Annotierten Textkorpora.* PhD thesis, IMS, University of Stuttgart, Stuttgart, Germany, 2002.

Li, Hang. Word clustering and disambiguation based on co-occurrence data. *Natural Language Engineering*, 8(1):25–42, 2002.

Lin, Dekang. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114, 1998.

Lynn, Teresa, Jennifer Foster, Mark Dras, and Elaine Uí Dhonnchadha. Active learning and the irish treebank. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, pages 23–32, Dunedin, New Zealand, December 4–6 2012.

MacQueen, Jacob. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, California, 1967. University of California Press.

Mahootiyan, Shahrzad. *Persian.* Routledge, 1997.

Maier, Wolfgang and Sandra Kübler. Are all commas equal? Detecting coordination in the Penn Treebank. In *Proceedings of the 12th International Workshop on Treebanks and Linguistic Theories*, pages 121–132, Sofia, Bulgaria, 2013.

Malouf, Robert, John Carroll, and Ann Copestake. Efficient feature structure operations without compilation. *Natural Language Engineering*, 6(1):29–46, 2000.

Manning, Christopher D. and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, MA, USA, 1999.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2), 1993.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the Association for Computational Linguistics*, pages 91–98, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

Megerdoomian, Karine. A computational analysis of the Persian noun phrase. Memoranda in Computer and Cognitive Science MCCS-00-321, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 2000.

Megerdoomian, Karine. Developing a Persian part of speech tagger. In *Proceedings of the 1st Workshop on the Persian Language and Computer*, pages 99–105, University of Tehran, Iran, 2004.

Mel'čuk, Igor A. *Dependency syntax: Theory and practice.* State University of New York, Albany, 1988.

MeshkatoDini, Mehdi. *dasture zabāne fārsi bar pāyeye nazariyeye gaštāri ["Introduction to Persian Transformational Syntax"].* Ferdowsi University Press, Mashad, Iran, 2 edition, 2001.

Miller, Scott, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–342. Association for Computational Linguistics, 2004.

Mirroshandel, SeyyedAbolghasem and Gholamreza GhassemSani. Unsupervised grammar induction using a parent based constituent context model. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 293–297, Amsterdam, The Netherlands, The Netherlands, 2008.

Mitchell, Melanie. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, USA, 1998.

Miyao, Yusuke. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model.* PhD thesis, University of Tokyo, 2006.

Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, pages 684–693, Berlin, Heidelberg, 2005. Springer-Verlag.

Momtazi, Saeedeh and Dietrich Klakow. A word clustering approach for language model-based sentence retrieval in question answering systems. In *Proceedings of the Annual International ACM Conference on Information and Knowledge Management*, pages 1911–1914. ACM, 2009.

Momtazi, Saeedeh, Hosein Sameti, Maryam FazelZarandi, and Mohammad Bahrani. Robust parsing for word lattices in continuous speech recognition systems. In *Proceedings of the 9th International Symposium on Signal Processing and Its Applications*, pages 1–4, February 12–15 2007.

Montague, Richard. *Formal Philosophy; Selected Papers of Richard Montague*. New Haven,Yale University Press, 1974.

Montazeri, Niloofar, Gholamreza GhassemSani, and Hossein Sameti. A fast and robust parser based on the Viterbi algorithm. In *Proceedings of the 11th Computer Society of Iran Conference*, Sharif University of Technology, Tehran, Iran, 2006.

Montemagni, Simonetta, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. Building the Italian syntactic-semantic treebank. In *In Abeillé (Abeillé, 2003), chapter 11*, pages 189–210. Kluwer, 2003.

Morita, Kazuhiro, El-Sayed Atlam, Masao Fuketra, Kazuhiko Tsuda, Masaki Oono, and Junichi Aoe. Word classification and hierarchy using co-occurrence word information. *Information Processing and Management*, 40(6):957–972, 2004.

Moyne, John and Guy Carden. Subject reduplication in Persian. *Linguistic Inquiry*, 5(2):205–249, 1974.

Moyne, John Abdel. The so-called passive in Persian. *Foundations of Language*, 12(2):249–267, 1974.

Müller, Stefan. Towards an HPSG analysis of Maltese. In Comrie, Bernard, Ray Fabri, Beth Hume, Manwel Mifsud, Thomas Stolz, and Martine Vanhove, editors, *Introducing Maltese Linguistics*, number 113 in Studies in Language Companion series, pages 83–112. John Benjamins Publishing Company, Amsterdam, Philadelphia, 2009.

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

Müller, Stefan. The CoreGram project: A brief overview and motivation. In Duchier, Denys and Yannick Parmentier, editors, *Proceedings of the Workshop on High-level Methodologies for Grammar Engineering*, pages 93–104, 2013.

Müller, Stefan. HPSG – A synopsis. In Alexiadou, Artemis and Tibor Kiss, editors, *Syntax – Ein internationales Handbuch zeitgenössischer Forschung*, Handbücher zur Sprach- und Kommunikationswissenschaft. Walter de Gruyter Verlag, Berlin, 2 edition, To Appear. URL `http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-hsk.html`.

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

Müller, Stefan and Janna Lipenkova. Serial verb constructions in Mandarin Chinese. In Müller, Stefan, editor, *Proceedings of the 16th International Conference on Head-driven Phrase Structure Grammar*, pages 234–254, Stanford, California, 2009. CSLI Publications.

Müller, Stefan and Bjarne Ørsnes. Positional expletives in Danish, German, and Yiddish. In Müller, Stefan, editor, *Proceedings of the 18th International Conference on Head-Driven Phrase Structure Grammar*, pages 167–187, Stanford, 2011. CSLI Publications.

Ngai, Grace and David Yarowsky. Rule writing or annotation: Cost-efficient resource usage for base noun chunking. In *Proceedings of the Association for Computational Linguistics*, pages 117–125, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

Niknejad, Seyed Ali. *tarrāhi va sāxte yek nemune motarjeme māšiniye fārsi be engelisi ["Design and Development of a Persian to English Translator Prototype"].* Master's thesis, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, 2008.

Nivre, Joakim. Dependency grammar and dependency parsing. *Journal Of The International Linguistic Association*, 2005a.

Nivre, Joakim. *Inductive Dependency Parsing of Natural Language Text.* PhD thesis, Växjö University, 2005b.

Nivre, Joakim and Béata Bandmann Megyesi. Bootstrapping a Swedish treebank using cross-corpus harmonization and annotation projection. In *Proceedings of Treebanks and Linguistic Theories*, 2007.

Nivre, Joakim, Johan Hall, and Jens Nilsson. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 2216–2219, 2006.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Nojoumian, Peyman. Modeling Persian language syntax and morphology in LingBench IDE^{TM}. Internship technical report, Faculty of Applied Sciences, Katholieke Universiteit Leuven, 2003. URL `http://www.teachmepersian.com/personal/Internship_Nojoumian2.pdf`.

Oepen, Stephan. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany, 2001.

Oepen, Stephan and Ulrich Callmeier. Measure for measure: Parser cross-fertilization. Towards increased component comparability and exchange. In *Proceedings of the 6th International Workshop on Parsing Technology*, pages 183–194, 2000.

Oepen, Stephan, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods: A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation*, 2(4):575–596, 2004.

Oroumchian, Farhad, Ehsan Darrudi, Fattane Taghiyareh, and Neeyaz Angoshtari. Experiments with Persian text compression for web. In *Proceedings of the 13th International World Wide Web Conference*, New York, USA, 2004.

Ørsnes, Bjarne. Preposed sentential negation in Danish. In Müller, Stefan, editor, *Proceedings of the 16th International Conference on Head-driven Phrase Structure Grammar*, pages 255–275, Stanford, California, 2009. CSLI Publications.

Osborne, Miles and Jason Baldridge. Ensemble-based active learning for parse selection. In Susan Dumais, Daniel Marcu and Salim Roukos, editors, *Proceedings of the Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics*, pages 89–96, Boston, Massachusetts, USA, 2004.

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

Osenova, Petya and Kiril Simov. BTB-TR05: BulTreeBank Stylebook–BulTreeBank Version 1.0. Technical report, Linguistic Modeling Laboratory, Bulgarian Academy of Sciences, Sofia, Bulgaria, 2004.

Pal, Santanu and Sivaji Bandyopadhyay. Bootstrapping method for chunk alignment in phrase based SMT. In *Proceedings of the Joint Workshop on Exploiting Synergies between Information Retrieval and Machine Translation and Hybrid Approaches to Machine Translation*, pages 93–100, Avignon, France, April 2012. Association for Computational Linguistics.

Parekh, Rajesh G. and Vasant Honavar. *Automata Induction, Grammar Inference, and Language Acquisition*, chapter 29, pages 746–785. Handbook of Natural Language Processing. Marcel Dekker, New York, 2000.

Penn, Gerald. Balancing clarity and efficiency in typed feature logic through delaying. In *Proceedings of the Association for Computational Linguistics*, pages 239–246, 2004.

Perrier, Guy. Interaction grammars. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 600–606, 2000.

Persian Academy of Language and Literature, . *dasture xatte fārsi [Grammar of Persian Orthography]*. Decisions of the Persian Academy of Language and Literature. Persian Academy of Language and Literature, Tehran, Iran, 2005.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the Association for Computational Linguistics*, pages 433–440, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Pighin, Daniele and Alessandro Moschitti. Efficient linearization of tree kernel functions. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, Boulder, Colorado, 2009. Association for Computational Linguistics.

Plaehn, Oliver and Thorsten Brants. Annotate - An efficient interactive annotation tool. In *6th Applied Natural Language Processing Conference*, Seattle, Washington, USA, 2000.

Pollard, Carl J. and Ivan A. Sag. *Information-Based Syntax and Semantics.* Number 13 in CSLI Lecture Notes. CSLI Publications, Stanford, California, 1987.

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

Pouramini, Ahmad and Elham Moridi. Annotation of grammatical function in the Persian treebank. *Procedia - Social and Behavioral Sciences: The 4th International Conference of Cognitive Science*, 32:302–307, 2012.

Pouramini, Ahmad and Nase Mozayani. An annotation scheme for a Persian treebank. In *Proceedings of Computational Linguistics In the Netherlands*, 2007.

RaisGhasem, Mohsen. *pardāzeše zabāne tabi?i va pardāzeše zabāne fārsi ["Natural Language Processing and Processing of the Persian Language"].* Master's thesis, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, 1991.

Rasooli, MohammadSadegh, Amirsaeid Moloodi, Manouchehr Kouhestani, and Behrouz MinaeiBidgoli. Syntactic valency lexicon for Persian verbs: The first steps towards Persian dependency treebank. In *Proceedings of the 5th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 227–231, Pozna n, Poland, June 2011.

Rasooli, MohammadSadegh, Manouchehr Kouhestani, and Amirsaeid Moloodi. Development of a Persian syntactic dependency treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 306–314, Atlanta, Georgia, 2013.

Ratnaparkhi, Adwait. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

Rezaei, Siamak. *Constraint-based Parsing of a Free Word Order Language: Persian.* Master's thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, UK, 1993.

Rezaei, Siamak. Parsing scrambling with path set: A graded grammaticality approach. In *Proceedings of the 6th International Workshop on Parsing Technologies*, 2000.

Rokach, Lior and Oded Maimon. *Clustering Methods*, chapter 15, pages 321–352. Data Mining and Knowledge Discovery Handbook. Springer, USA, 2005.

Rosenblatt, Frank. The perceptron – A perceiving and recognizing automaton. Report 85–460–1, Cornell Aeronautical Laboratory, 1957.

Sajjadi, Armin and Ahmad AbdollahzadeBarforoush. tahlile nahviye zabāne fārsi be komake gerāmere peyvandi [“Syntactic analysis of the Persian language by Link grammar”]. *pardāzeše alāyem va dādehā [Signal and Data Processing]*, 1:25–40, 2009.

Salehi, Bahar, Narjes Askarian, and Afsaneh Fazly. Automatic identification of Persian light verb constructions. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 201–210, Berlin, Heidelberg, 2012. Springer-Verlag.

Sampson, Geoffrey. A proposal for improving the measurement of parse accuracy. *International Journal of Corpus Linguistics*, 5:53–68, 2000.

Sampson, Geoffrey. *Thoughts On Two Decades Of Drawing Trees*, chapter 2, pages 23–41. Treebanks: Building and Using Parsed Corpora. Kluwer Academic Publishers, The Netherlands, 2003.

Samuelsson, Christer. Morphological tagging based entirely on Bayesian inference. In Eklund, R., editor, *Proceedings of the 9th Scandinavian Conference on Computational Linguistics*, pages 225–238, Stockholm, Sweden, 1994.

Samuelsson, Christer and Wolfgang Reichl. A class-based language model for large-vocabulary speech recognition extracted from part-of-speech statistics. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE Computer Society, 1999.

Samuelsson, Yvonne and Martin Volk. Phrase alignment in parallel treebanks. In *Proceedings of Treebanks and Linguistic Theories*, pages 91–102, 2006.

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

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

Sanamrad, MohammadAli and Haruy Matsumoto. PERSIS: A natural-language analyzer for Persian. *Journal of Information Processing*, 8(4):271–279, 1986.

Sarabi, Zahra and Morteza Analouie. A new DOP model for phrase-structure parsing of Persian sentences. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 45–54, 2012.

Sarabi, Zahra, Hooman Mahyar, and Mojgan Farhoodi. ParsiPardaz: Persian language processing toolkit. In *Proceedings of the 3rd International eConference on Computer and Knowledge Engineering*, pages 79–85, Ferdowsi University of Mashhad, Mashhad, Iran, 2013.

Schabes, Yves. *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, University of Pennsylvania, 1990.

Schank, Roger C. *Conceptual Information Processing.* Elsevier Science Inc., New York, NY, USA, 1975.

Schluter, Natalie. *Treebank-based Deep Grammar Acquisition for French Probabilistic Parsing Resources.* PhD thesis, Dublin City University, 2011.

Sennrich, Rico, Gerold Schneider, Martin Volk, and Martin Warin. A new hybrid dependency parser for German. In Chiarcos, C., R. E. Castilhode , and M. Stede, editors, *Von der Form zur Bedeutung: Texte automatisch verarbeiten "[From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009]"*, pages 115–124, Tübingen, 2009.

Seraji, Mojgan, ata Megyesi Be and Joakim Nivre. Bootstrapping a Persian dependency treebank department. *Linguistic Issues in Language Technology*, 7(18), 2012a.

Seraji, Mojgan, Beáta Megyesi, and Joakim Nivre. A basic language resource kit for Persian. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2245–2252, Istanbul, Turkey, 2012b. European Language Resources Association (ELRA).

Seraji, Mojgan, Beata Megyesi, and Joakim Nivre. Dependency parsers for Persian. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 35–43, 2012c.

Settles, Burr. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Settles, Burr. *Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan & Claypool Publishers, 2012. URL `http://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018`.

Seung, H. Sebastian, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 287–294, New York, NY, USA, 1992. ACM.

Shaghaghi, Vida. kamminamāhāye zabāne fārsi ["Persian quantifiers"]. *majall-eye adabiyāt va olume ensāni [Journal of Literature and Humanities]*, 35(3): 621–650, 2002.

Shamsfard, Mehrnoush, Hoda Sadat Jafari, and Mahdi Ilbeygi. STeP-1: A set of fundamental tools for Persian text processing. In Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 859–865, Valletta, Malta, May 19–21 2010. European Language Resources Association (ELRA).

Shannon, Claude E. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

SharifiAtashgah, Masood and Mahmood Bijankhan. Corpus-based analysis for multi-token units in Persian. In *Third Workshop on Computational Approaches to Arabic Script-based Languages [at] MT*, Ottawa, Canada, 2009.

SharifiAtashgah, Massoud. *toulide nimexodkāre deraxtbānke goruhhāye nahvi dar motune fārsi ["Semi-automatic Generation of Treebanks in Persian Texts"].* PhD thesis, University of Tehran, 2009.

Simov, Kiril. Grammar extraction from an HPSG corpus. In *Proceedings of the Recent Advances in Natural Language Processing Conference*, pages 285–287, Tzigov Chark, Bulgaria, 2001.

Simov, Kiril. 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, 2002.

Simov, Kiril and Petya Osenova. CLaRK system: Construction of treebanks. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 183–198, 2002.

Simov, Kiril and Petya Osenova. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, pages 17–24, 2003.

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

Simov, Kiril, Gergana Popova, and Petya Osenova. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In Wilson, Andrew, Paul Rayson, and Tony McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pages 135–142, 2002a.

Simov, Kiril, Gergana Popova, and Petya Osenova. *HPSG-based syntactic treebank of Bulgarian (BulTreeBank)*, chapter 13, pages 135–142. Lincom GmbH, Muenchen, 2002b.

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

Simov, Kiril, Petya Osenova, Alexander Somov, and Milen Kouylekov. Design and implementation of the Bulgarian HPSG-based treebank. *Research on Language and Computation*, 2:495–522, 2004.

Simov, Kiril Ivanov. HPSG-based annotation scheme for corpora development and parsing evaluation. In Nicolov, Nicolas, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing III*, volume 260, pages 327–336. John Benjamins, Amsterdam/Philadelphia, 2003.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 88–95, Stroudsburg, PA, USA, 1997.

Skut, Wojciech, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. A linguistically interpreted corpus of german newspaper text. In *In Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, pages 705–711, 1998.

Sleator, Daniel D. K. and Davy Temperley. Parsing English with a link grammar. In *Proceedings of the 3rd International Workshop on Parsing Technologies*, 1993.

SoheiliIsfahani, Abulghasem. *Noun Phrase Complementation in Persian.* PhD thesis, University of Illinois at Urbana-Champaign, 1976.

Steedman, Mark. Categorial grammar. *Lingua*, 90:221–258, 1993.

Steedman, Mark, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. Example selection for bootstrapping statistical parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 157–164, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

Stolcke, Andreas. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, 2002.

Stump, Gregory T. *Inflectional Morphology: A Theory of Paradigm Structure.* Cambridge Studies in Linguistics. Cambridge University Press, 2001.

Tabatabayi, Alaeddin. fe'le morakkab dar zabāne fārsi ["Compound verb in Persian"]. *Nāmeye Farhangestān [Academia's letter]*, 26:26–34, 2005.

Tabibzadeh, Omid. *sāxte zabāne fārsi: bar asāsenazariyeye goruhhāye xodgardān dar dasture vābastegi ["Persian Grammar: A Theory of Autonomous Phrases Based on Dependency Grammar"].* Našr-e Markaz [Markaz Publication], Tehran, Iran, 2012.

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

Taleghani, Azita H. *Modality, Aspect and Negation in Persian.* John Benjamins Publishing, Amsterdam/Philadelphia, 2008.

Taslimipoor, Shiva, Afsaneh Fazly, and Ali Hamzeh. Using noun similarity to adapt an acceptability measure for Persian light verb constructions. In Calzolari, Nicoletta, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Istanbul, Turkey, May 23–25 2012. European Language Resources Association.

Tateisi, Yuka, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. Translating the XTAG English grammar to HPSG. In *The 4th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 172–175, 1998.

Taylor, Ann, Mitchell Marcus, and Beatrice Santorini. *The Penn Treebank: An Overview*, chapter 1, pages 5–22. Treebanks: Building and Using Parsed Corpora. Kluwer Academic Publishers, The Netherlands, 2003.

Teixeira, Jorge, Luís Sarmento, and Eugénio Oliveira. A bootstrapping approach for training a NER with conditional random fields. In *Proceedings of the 15th Portugese Conference on Progress in Artificial Intelligence*, pages 664–678, Berlin, Heidelberg, 2011. Springer-Verlag.

Teleman, Ulf. *Manual för grammatisk beskrivning av talad och skriven svenska.* Lundastudier i nordisk språkvetenskap. Studentlitteratur, Lund, 1974.

Thelen, Michael and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Association for Computational Linguistics Conference on Empirical Methods in Natural Language Processing*, pages 214–221, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Thompson, Cindi, Roger Levy, and Christopher Manning. A generative mode for FrameNet semantic role labeling. In Lavrac, Nada, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *Proceedings of the 14th European Conference on Machine Learning*, volume 2837 of *Lecture Notes in Computer Science*, pages 397–408, Cavtat-Dubrovnik, Croatia, 2003. Springer.

Tinsley, John, Mary Hearne, and Andy Way. Exploiting parallel treebanks to improve phrase-based statistical machine translation. In *The Sixth International Workshop on Treebanks and Linguistic Theories*, 2007.

Tounsi, Lamia, Mohammed Attia, and Josef Genabithvan . Automatic treebank-based acquisition of Arabic LFG dependency structures. In *Proceedings of the European Chapter of the Association for Computational Linguistics 2009, Workshop on Computational Approaches to Semitic Languages*, pages 45–52, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Toutanova, Kristina and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63–70, Hong Kong, October 2000.

Uibo, Heli, Krista Liin, and Martin Volk. Phrase alignment of Estonian-German parallel treebanks. In *Proceedings of the Workshop on Exploiting parallel corpora in up to 20 languages*, 2005.

Uszkoreit, Jakob and Thorsten Brants. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of the International Conference of the Association for Computational Linguistics.* Association for Computational Linguistics, 2008.

VahediLangaroodi, MohammadMehdi. sāxthāye fe'liye majhul bā fe'le "šodan" dar zabāne fārsi ["Verbal passive constructions with the verb "be" in the Persian language"]. *Modarrese olume ensāni [Human Sciences Modarres]*, 7:75–101, 1999. URL `http://www.noormags.com/view/fa/articlepage/50037`.

Viterbi, Andrew J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2):260–269, 1967.

Vlachos, Andreas. A stopping criterion for active learning. *Journal of Computer, Speech and Language*, 22(3):295–312, 2008.

Volk, Martin and Yvonne Samuelsson. Bootstrapping parallel treebanks. In Hansen-Schirra, Silvia, Stephan Oepen, and Hans Uszkoreit, editors, *COLING 2004 5th International Workshop on Linguistically Interpreted Corpora*, pages 63–70, Geneva, Switzerland, 2004.

Wahlster, Wolfgang, editor. *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer Verlag, Berlin, 2000.

Xia, Yunqing, Boyi Hao, and Kam-Fai Wong. Opinion target network and bootstrapping method for Chinese opinion target extraction. In *Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology*, pages 339–350, Berlin, Heidelberg, 2009. Springer-Verlag.

Yoshinaga, Naoki and Yusuke Miyao. Grammar conversion from ltag to hpsg. In *Proceedings of the Sixth ESSLLI Student Session*, pages 309–324, 2002.

Younger, Daniel H. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

Yu, Kun, Miyao Yusuke, Xiangli Wang, Takuya Matsuzaki, and Junichi Tsujii. Semi-automatically developing Chinese HPSG grammar from the Penn Chinese treebank for deep parsing. In *Proceedings of the International Conference on Computational Linguistics*, pages 1417–1425, Beijing, China, 2010.

Zhang, Cha and Tsuhan Chen. An active learning framework for content-based information retrieval. In *IEEE Transactions on Multimedia*, volume 4, pages 260–268, 2002.

Zhechev, Ventsislav and Andy Way. Automatic generation of parallel treebanks. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1105–1112, 2008.

Zhu, Jingbo and Eduard Hovy. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 783–790, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Zhu, Jingbo and Huizhen Wang. Learning a stopping criterion for active learning for word sense disambiguation and text classification. In *Proceedings of the 3rd IJNLP*, pages 366–372, Heydarabad, India, 2008.

Zhu, Jingbo, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1137–1144, 2008b.

Zinsmeister, Heike, Jonas Kuhn, Bettina Schrader, and Stefanie Dipper. TIGER Transfer - From LFG structures to the TIGER Treebank. Technical report, IMS, University of Stuttgart, 2001.