

VOLKAN ESGEL

SEMANTIC ROLE LABELING FOR TURKISH PROPBANK

M.S. Thesis

VOLKAN ESGEL

2019

IŞIK UNIVERSITY

2019

SEMANTIC ROLE LABELING FOR TURKISH PROPBANK

VOLKAN ESGEL

B.S., Computer Engineering, IŞIK UNIVERSITY, 2012

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY

2019

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

SEMANTIC ROLE LABELING FOR TURKISH PROPBANK

VOLKAN ESGEL

APPROVED BY:

Prof. Dr. Olcay Taner YILDIZ Işık University _____
(Thesis Supervisor)

Assist. Prof. İlknur KARADENİZ Işık University _____
EROL

Assoc. Prof. Arzucan ÖZGÜR Boğaziçi University _____

APPROVAL DATE: / /

SEMANTIC ROLE LABELING FOR TURKISH PROPBANK

Abstract

People's communication with each other takes place through sentences that combine words with different purposes. Words can gain different meanings with the presence of other words in the sentences in which they take place.

With the rapid development of technology, the studies on understanding of human language by computational power have gained speed. These studies are generally referred to Natural Language Processing and their main purpose is to understand the sentences in human communication.

The words in the sentence fulfill different purposes. Some words describe an event, while other words indicate details of that event. Defining the semantic roles of words is possible with different algorithms.

This study first started by contributing to the process of determining the semantic roles of the word groups in the sentence by manpower. In addition, the semantic roles in the English sentences were parsed and shared on a web site with the marked roles in the Turkish sentences for comparison purposes. Finally, it is tried to measure how the algorithms aiming to find the semantic roles of the words in the sentence perform automatically for Turkish.

Keywords: Natural Language Processing, PropBank, Semantic Role Labeling

TÜRKÇE PROPBANK İÇİN ANLAMSAL ROL ETİKETLEMESİ

Özet

İnsanların birbirleriyle olan iletişimleri farklı amaçlardaki kelimelerin birleştiği cümleler aracılığı ile gerçekleşmektedir. Kelimeler, yer aldıkları cümlelerde diğer kelimelerin de varlığı ile farklı anlamlar kazanabilmektedirler.

Teknolojinin hızla gelişmesiyle birlikte, hesaplamalı güçler tarafından insan dilini anlama çalışmaları hız kazanmıştır. Bu çalışmalar genel olarak Doğal Dil İşleme olarak anılmaktadır ve asıl amaçları insan iletişiminde yer alan cümlelerin anlaşılabilmesidir.

Cümle içerisinde yer alan kelimeler farklı amaçları yerine getirmektedirler. Bazı kelimeler bir olayı anlatırken, diğer kelimeler bu olaya ait detayları belirtmektedirler. Kelimelerin anlamsal rollerinin tanımlanması ise farklı algoritmalar ile mümkün olabilmektedir.

Bu çalışma ilk olarak cümlede yer alan kelime gruplarına ait anlamsal rollerin insan gücü ile belirlenmesi işlemine katkı sağlanarak başlamıştır. Ayrıca doğal dil işleme ile ilgili olarak Türk dilinde yapılacak çalışmalarda karşılaştırma amaçlı olarak kullanılmak üzere İngilizce cümlelerde yer alan anlamsal roller ayrıştırılmış ve Türkçe cümlelerdeki işaretlenmiş roller ile birlikte bir web sitesi üzerinden açık şekilde paylaşılmıştır. Son olarak ise cümledeki kelimelerin anlamsal rollerini otomatik olarak bulmayı amaçlayan algoritmaların Türkçe için nasıl performans gösterdiği ölçülmeye çalışılmıştır.

Anahtar kelimeler: Doğal Dil İşleme, PropBank, Anlamsal Rol Etiketleme

Acknowledgements

First of all, I would like to express my gratitude to my thesis supervisor Prof. Dr. Olcay Taner YILDIZ for all of his guidance on all stages of this master thesis. Without his advice and guidance, I do not think this work would have been possible. I will always remember his patience, understanding and kind personality.

Finally, my sincere thanks to my family, especially my brother Hakan, for their endless and peerless support.

To my family...

Table of Contents

Abstract	ii
Özet	iii
Acknowledgements	iv
List of Tables	viii
List of Figures	ix
List of Abbreviations	x
1 Introduction	1
2 Literature Review	4
2.1 Penn Treebank Project	4
2.1.1 Tagset for Part-of-Speech	4
2.1.2 Tagging Process for Part-of-Speech	5
2.2 Semantic Role Labeling	6
2.2.1 Calibrating Features	7
2.2.2 The PropBank and Semantic Role Labeling	7
2.2.3 Semantic Role Tagging	9
2.2.4 System Architecture	9
2.2.5 Pruning Algorithm	10
3 PropBank Data	11
3.1 Parse of English PropBank Data	11
3.2 Assignment of Predicate and Arguments in Turkish PropBank . .	14
3.2.1 Selection of Predicates in Turkish PropBank Toolkit	14
3.2.2 Selection of Arguments in Turkish PropBank Toolkit	16
4 PropBank Website	18
4.1 Preparation of the PropBank Website	18
4.2 Technologies Used in the Study	21

5	Features	23
5.1	Word-Based Features	23
5.1.1	Predicate	23
5.1.2	Phrase Type	23
5.1.3	Voice	24
5.1.4	Named Entities in Constituents	24
5.2	Tree-Based Features	25
5.2.1	Path	25
5.2.2	Position	25
5.2.3	Sub-categorization	26
6	Classifiers	28
6.1	Rocchio	28
6.2	Naive Bayes	29
6.3	kNN	30
6.4	Linear Perceptron	32
6.5	Multi Layer Perceptron	33
7	Experiments	34
7.1	Setup	34
7.2	Prediction of the Predicates	35
7.3	Prediction of the Arguments	37
8	Conclusion	42
	Reference	43

List of Tables

2.1	List of Part-of-Speech Tagset for The Penn Treebank.	5
3.1	Details of the parsed JSON data format.	14
4.1	Color schema for Semantic Role Labels.	20
4.2	Web Application Paths for AngularJS.	22
7.1	Setup Files with Detailed Predicate Information.	34
7.2	Number of class labels in Turkish PropBank.	34
7.3	Result of first experiment to detect predicates.	36
7.4	Result of second experiment to detect predicates : Reverse all of the leaf nodes.	37
7.5	Result of third experiment to detect predicates : Added examina- tion of multiple predicates.	37
7.6	Argument Prediction Results for different window sizes.	38
7.7	Confusion Matrix for the Naive Bayes classifier executed with win- dow size 3.	40
7.8	Confusion Matrix for the Linear Perceptron classifier executed with window size 3.	41

List of Figures

2.1	Sample text with POS tags - before the correction step.	6
2.2	Sample text with POS tags - after the correction step.	6
2.3	Pruning.	10
3.1	An example HTML code for the list items with verb and sense ID.	12
3.2	Sentences with labeled arguments in Rochester Data [5].	12
3.3	Temporary data in JSON format for the parsed data.	13
3.4	A screenshot of the PropBank Predicate Editor Tool.	15
3.5	Selecting Predicates in PropBank Predicate Editor Tool.	15
3.6	A sample screenshot for the PropBank Argument Editor Tool.	16
3.7	Selecting Arguments in PropBank Argument Editor Tool.	17
4.1	Homepage - List of Predicates found in Turkish PropBank.	19
4.2	Page - Predicate Details.	19
4.3	Page - Show Tree in Predicate Details.	21
5.1	Lexicalization of phrase type features.	24
5.2	Sample tree for the path feature.	25
5.3	Illustration of the position feature.	26
5.4	Subcategorization example for the predicate <i>open</i>	27
6.1	Rochhio Classification [9].	29
7.1	Sample feature data extracted from Turkish PropBank.	39

List of Abbreviations

NLP	N atural L anguage P rocessing
PropBank	The P roposition B ank
SL	S emantic R ole
SRL	S emantic R ole L abeling
ARG	A rgument
NE	N amed E ntity
NER	N amed E ntity R ecognition
POS	P art of S peech
GUI	G raphical U ser I nterface
JSON	J ava S cript O bject N otation
HTML	H yper T ext M arkup L anguage
REST	R epresentational S tate T ransfer
ISO	The I nternational O rganization for S tandardization
ID3	I terative D ichotomiser 3
kNN	k Nearest Neighbors
MLP	M ulti L ayer P erceptron
ANN	A rtificial N eural N etwork
LDA	L inear D iscriminant A nalysis
QDA	Q uadratic D iscriminant A nalysis

Chapter 1

Introduction

Natural Language Processing refers to the capability of understanding natural human languages, like Turkish and English, using computer systems. To make this possible, combination of linguistics and current Machine Learning systems is used. This is simply a process like diagramming sentences in primary school. Accordingly, there are several stages defined to realize this purpose including tokenization, parsing, stemming, part-of-speech tagging, determination of the language and semantic relationships.

PropBank is a corpus of text annotated with the arguments of each predicate. Arguments have different semantic relations with each predicate and all of the arguments are grouped with their semantic roles. PropBank also includes the related information for different senses of predicates. It is so important because predicates generally requires different type of arguments due to their senses in the sentence. For this reason, each meaning of the predicate is labeled with a sense ID. For each senses of predicates, there are also individual frameset files that represents the expected structure of arguments.

Semantic Role is defined as a connection between a component and a participant in a sentence. There are different type of semantic roles like AGENT, PATIENT, INSTRUMENT.

Semantic Role Labeling is defined as a computational identification task that determines the semantic roles of each argument for the each predicate found in a sentence. In SRL, automated algorithms assign labels to constituents with semantic roles. In the current approaches, supervised machine learning algorithms are widely used to accomplish this task. This requires large amounts of manually generated data for training and test cases. FrameNet and PropBank resources also have a common usage in the different stages of this task.

Although common methods are used in studies for different languages, differences in the structures of natural human languages require that all these studies be performed separately for each language.

In this study, we present the study about Semantic Role Labeling for Turkish language. In the beginning, we contributed the manual annotation of the required PropBank dataset for Turkish language [1] which is used to train and test our SRL study. Then, we tried to measure the success of these methods for Turkish language by training and testing our system by using different types of features and classifiers. Additionally, with the automated parsing of English PropBank data, we prepared a website which includes Turkish and English representations of the same sentences with the semantically annotated labels.

The key contributions of the study are shown in the order in which they are located throughout the thesis.

1. Contribution for the manual annotation of a Turkish PropBank [1] data which consist of the translation of English Penn-TreeBank.
2. Parsing of English PropBank data with semantic roles which is used to test related works.
3. Development of Turkish PropBank website that shows the English and Turkish representations of sentences with semantic role labels and annotated tree structures.

The parts of the thesis are located as follows.

In Chapter 2, we introduce the general concepts required to know for semantic role labeling. You can also find the detailed information about the Penn Treebank Project, PropBank and Semantic Role Labeling in this section.

In Chapter 3, we provide detailed information about the Automatic Semantic Role Labeling work in Turkish language, which is the main aim of this study. As we know, this is the first Automatic SRL study for the Turkish.

In Chapter 4, we present the details of our study on Turkish PropBank Website which is available in public.

In Chapter 5, we give explanations of the features which is used to assign related semantic role labels to the constituents. They are placed under two sub-sections according to their *word* or *tree-based* structure.

In Chapter 6, we present the classifiers used to automatic prediction of semantic role labels along with short summaries.

In Chapter 7, we detail the result of our experiments with discussions.

We conclude our work in Chapter 8.

Chapter 2

Literature Review

2.1 Penn Treebank Project

With the rapid development of technology, research has been accelerated for the understanding of human language by computers. Penn Treebank Project [2] provides the required annotated corpus that the researchers need to study on areas like natural language processing, recognition of speech, integrated spoken language technologies, and theoretical linguistics. During the development of text and speech understanding systems, automatic extraction of language information from a very large corpora plays a significant role.

The corpus of the Penn Treebank involves over 4.5 million American English words. In the early years of the project, most of the skeletal syntactic structure and Part-of-speech (POS) data is annotated for the corpus. This corpus data is available and can be accessible for the members of the Linguistic Data Consortium.

2.1.1 Tagset for Part-of-Speech

The Penn Treebank consist of 48 tags in all. While 36 tags are reserved for the POS tags, 12 additional tags refers to the currency and punctuation symbols. A comprehensive explanation of the tagset rules can be found in Santorini (1990) [3].

Table 2.1: List of Part-of-Speech Tagset for The Penn Treebank.

CC	Coordinating conjunction	UH	Interjection
CD	Cardinal number	VB	Verb, base form
DT	Determiner	VBD	Verb, past tense
EX	Existential there	VBG	Verb, gerund/present participle
FF	Foreign word	VBN	Verb, past participle
IN	Preposition/subordinating participle conjunction	VBP	Verb, non-3rd ps. sing. present
JJ	Adjective	VBZ	Verb, 3rd ps. sing. present
JJR	Adjective, comparative	WDT	wh-determiner
JJS	Adjective, superlative	WP	wh-pronoun
LS	List item marker	WP\$	Possessive wh-pronoun
MD	Modal	WRB	wh-adverb
NN	Noun, singular or mass	#	Pound sign
NNS	Noun, plural	\$	Dollar sign
NNP	Proper noun, singular	.	Sentence-final punctuation
NNPS	Proper noun, plural	,	Comma
PDT	Predeterminer	:	Colon, semi-colon
POS	Possessive ending	(Left bracket character
PRP	Personal pronoun)	Right bracket character
PP\$	Possessive pronoun	"	Straight double quote
RB	Adverb	'	Left open single quote
RBR	Adverb, comparative	"	Left open double quote
RBS	Adverb, superlative	'	Right close single quote
RP	Particle	"	Right close double quote
SYM	Symbol (mathematical or scientific)		
TO	To		

Note that one of the best known American English corpuses is Brown University Standard Corpus. A list of tagset for Penn Treebank is shown in Table 2.1 [2].

2.1.2 Tagging Process for Part-of-Speech

Penn Treebank's tagged version is created in two phases [2].

In the first phase, an automated stage is used to assign POS tags. In the earlier times of Penn Treebank Project, a stochastic algorithm which uses an altered tagset variant of the Brown Corpus predicts POS tagsets with 3-5 percent error

Battle-tested/NNP industrial/JJ managers/NNS here/RB always/RB
 buck/VB up/IN nervous/JJ newcomers/NNS with/IN the/DT tale/NN
 of/IN the/DT first/JJ of/IN their/PP\$ countrymen/NNS to/TO
 visit/VB Mexico/NNP ,/, a/DT boatload/NN of/IN samurai/NNS
 warriors/NNS blown/VBN ashore/RB 375/CD years/NNS ago/RB./.
 /From/IN the/DT beginning/NN ,/, it/PRP took/VBD a/DT man/NN
 with/IN extraordinary/JJ qualities/NNS to/TO succeed/VB in/IN
 Mexico/NNP ,/, / says/VBZ Kimihide/NNP Takimura/NNP ,/,
 president/NN of/IN Mitsui/NNS group/NN s/POS Kensetsu/NNP
 Engineering/NNP Inc./NNP unit/NN ./.

Figure 2.1: Sample text with POS tags - before the correction step.

Battle-tested/NNP*/JJ industrial/JJ managers/NNS here/RB always/RB
 buck/VB*/VBP up/IN*/RP nervous/JJ newcomers/NNS with/IN the/DT
 tale/NN of/IN the/DT first/JJ of/IN their/PP\$ countrymen/NNS to/TO
 visit/VB Mexico/NNP ,/, a/DT boatload/NN of/IN samurai/NNS*/FW
 warriors/NNS blown/VBN ashore/RB 375/CD years/NNS ago/RB./.
 /From/IN the/DT beginning/NN ,/, it/PRP took/VBD a/DT man/NN
 with/IN extraordinary/JJ qualities/NNS to/TO succeed/VB in/IN
 Mexico/NNP ,/, / says/VBZ Kimihide/NNP Takimura/NNP ,/,
 president/NN of/IN Mitsui/NNS*/NNP group/NN s/POS Kensetsu/NNP
 Engineering/NNP Inc./NNP unit/NN ./.

Figure 2.2: Sample text with POS tags - after the correction step.

rate. More recently, a cascade of stochastic and Penn Treebank tagset based rule-driven taggers realized the automatic POS assignments with error rate 2-6%. In Figure 2.1, a sample text with the automatically assigned POS tagsets before correction is presented.

Then, the results of the automatic POS tagging phase is corrected by the annotators. Figure 2.2 shows the sample text with POS tags after manual correction phase.

2.2 Semantic Role Labeling

Automatic, precise, and wide-inclusion methods that can explain normally happening content with semantic structure can assume a key job in NLP applications,

for example, data extraction, question replying, and rundown. One of the methods to generate such semantic structure is Semantic Role Labeling.

For each predicate found in a sentence, a semantic role labeler should recognize and label its semantic arguments. This method involves recognition of these semantic arguments for the word groups and assignment of particular labels to them.

2.2.1 Calibrating Features

Development of the statistical semantic analyzers has increased with the availability of semantically annotated corpora like Proposition Banks and FrameNet. In these systems, the syntactic components are annotated with semantic role labels mainly using syntactic information features for the syntactic parse tree. Xue and Palmer, 2004 [4], obtained a fairly significant improvements in their SRL tasks using some further features.

2.2.2 The PropBank and Semantic Role Labeling

To take the generalizations which are not sufficiently submitted in the treebank parse trees, a semantic annotation layer is added to the TreeBank II by the PropBank [4]. As an example, the following sentences are handled:

- (1) *John broke the window into a million pieces yesterday.*
- (2) *The window broke into a million pieces yesterday.*

In sentences (1) and (2), *the window* is in the same role related with the verb *break* although they are not in the same syntactic positions. In PropBank annotations, there are a set of fixed roles with separated labels for each verb. These roles are represented with integers in order starting with 0 and a prefix with ARG. For the *break* predicate, an example for the numbered arguments is given as following.

ARG0 the breaker

ARG1 thing broken

ARG2 instrument

ARG3 pieces

Note that numbers used to label the semantic roles is specific to the related verb and same numbers can be used for different predicates. An argument that has the same number for its label, e.g. ARG1, may have different semantic roles for different verbs. The numbered elements are known as core elements for the verb.

Besides the core elements, there are also some elements which have weaker relation with the verb. These elements have labels starting with ARGM prefix and then including a tag which shows the type of adjunct.

In the above sentences (1) and (2), *yesterday* does not have a direct relation with the verb *break* and it covers a broad range of verbs. For that reason, ARGM label with a -TMP secondary tag which shows the temporal state of the constituent will be used for its role in the sentence. In particular, secondary tags are a global classification of adjunct-like components. In the Proposition Bank, 12 secondary tags are found for ARGMs: DIR, LOC, MNR, TMP, REC, EXT, PRD, PRP, DIS, ADV, MOD, NEG2.

Some verbs require different sets of arguments for each meaning. Differentiating these senses is the first step for exact detection of the semantic roles of their arguments. Let's examine the verb "pass" as an example. When it is used in the meaning of "vote and pass", it takes three arguments, legislative body, bill and law. However, if it means "overtake", it takes only two arguments, entity moving forward and a falling entity. Each perception of this verb is likely to occur in a number of different frame sub-categories and is therefore referred to as a frameset.

2.2.3 Semantic Role Tagging

Depending on what kind of information you want to learn automatically, there are various methods for the semantic role labeling task on the basis of the PropBank annotation. Xue and Palmer [4] attempt to estimate the basic arguments of ARG[0-5] and the secondary tags for ARGMs. For each predicate found in the sentence, they determine the core arguments ARG[0-5], and the secondary tags for ARGMs. Note that semantic argument means both of numbered and ARGM arguments. They label the elements which have not a semantic relation with the given verb as NULL.

2.2.4 System Architecture

Although this could be deemed a multi-category classification problem, there are at all events two reasons why such a fundamental methodology does not function properly. The first is that the majority of components in a syntactic tree for a certain verb are not their semantical arguments. Existing machine learning algorithms will not make any effect on positive samples drowned by NULL tagged negative samples. The second and more obvious reason is that the information which succeeds in the separation of arguments from NULL values does not show the same success when distinguishing different types of arguments. Xue and Palmer [4] examine the following 3-stage architecture based on these thoughts:

Step 1: By the usage of a basic algorithm, components that do not have a clear semantic relationship with the predicate are filtered and training time is reduced.

Step 2: Then, the candidates obtained in the first step should be classified according to whether they are semantic arguments or not.

Step 3: In the last step, the components selected as arguments should be classified into one of the existing classes or NULL. This classification task is accomplished by the use of various category classifiers.

2.2.5 Pruning Algorithm

In the first step, Pruning Algorithm can be used as a simple algorithm to filter out useless components due to weak relation with the predicate. It is also presented in Figure 2.3.

Phase 1: Firstly, in a syntactic dependency tree, the predicate is determined as the current node. Then, all of the components found in the same level with the predicate are collected until these nodes are coordinated with the predicate. When a sibling is a PP, gather its instant children as well.

Phase 2: The syntactic head of the current node is identified as the new current node and this process is repeated until it reaches the root of the tree.

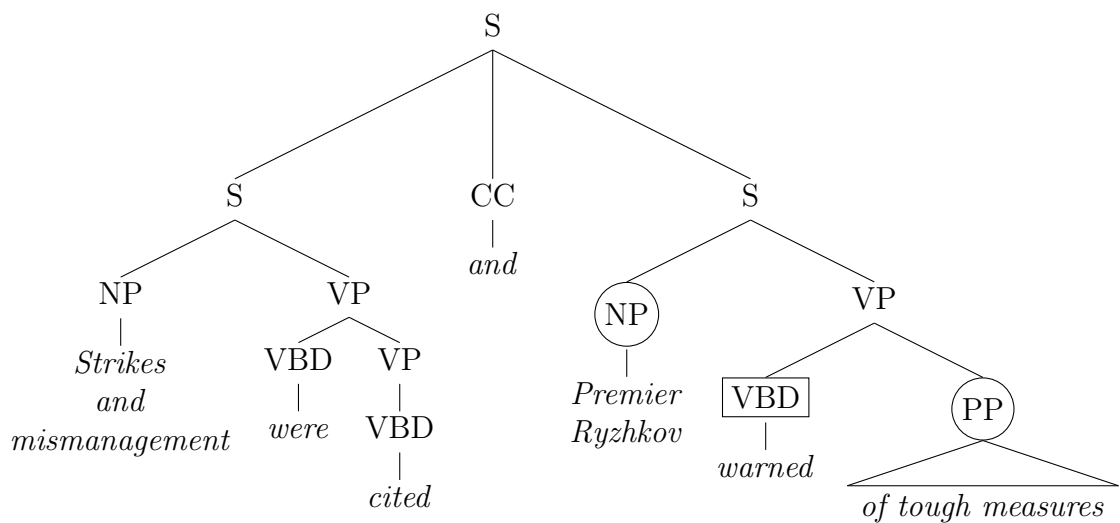


Figure 2.3: Pruning.

Chapter 3

PropBank Data

3.1 Parse of English PropBank Data

For NLP studies, we have PropBank data in Turkish which consists of the Turkish translations of the original PropBank sentences, which are written in English. We thought that it would be more useful to compare the studies conducted in Turkish language with the studies conducted in the original language of PropBank. Therefore, we extracted the arguments marked in the English language and translated them into the format of the PropBank data we use in the Turkish language. Note that the data was converted to JSON format temporarily. Then, it was converted to the format of PropBank data in Turkish with the study of another researcher. In this study, we used the Rochester Data [5] as a source to extract the marked arguments in the original language.

In the main page of the resource data [5], all of the verbs are given as separated list items and each list item has a link to the related page which contains the labeled sentences for the selected verb and sense ID. Within each list item with `li` HTML tag, there is a link which contains the dot separated verb and its related sense ID, in a notation like `verb.senseID`, between the `<a>` and `` HTML tags. The link which refers to the related verb and sense ID page is presented with `href` attribute. An example HTML code for the main page is given in Figure 3.1.

```

<ul>
  <li>
    <a href="a/abandon.01.html">abandon.01</a> 57
  </li>
  <li>
    <a href="a/abandon.02.html">abandon.02</a> 2
  </li>
</ul>

```

Figure 3.1: An example HTML code for the list items with verb and sense ID.

After this, the page which contains the example sentences with labeled words is called as detail page. When the parser is executed, it walks through the all of the detail pages for each verb and sense, and parse all of the labeled arguments. The dot separated verb and sense ID is given between `<h1>` and `</h1>` HTML tags in detail pages. All sentences are presented between `` and `` HTML tags, so each sentence is a list item of the unordered list. In each sentence, the labeled constituents are enclosed in square brackets. The name of the related argument is presented with `sub` tag, and the labeled item is placed between `` and `` tags. An example representation of the detail pages are given in Figure 3.2.

```

<h1>believe.01</h1>

<ul>
  <li>
    [<sub>ARG0</sub> <span class="ARG0"> I</span>] [<sub>rel
</sub> <span class="rel"> believe</span>] in [<sub>ARG1-in
</sub> <span class="ARG1"> the system</span>] .
  </li>
  <li>
    '' [<sub>ARG0</sub> <span class="ARG0"> Mr. Perkins</span
>] [<sub>rel</sub> <span class="rel"> believes</span>] , [<sub
>ARGM-DIS</sub> <span class="ARGM"> however</span>] , [<sub
>ARG1</sub> <span class="ARG1"> that the market could be
stabilized *-1 if California investor Marvin Davis steps back
in to the United bidding with an offer of $ 275 *U* a share
</span>] .
  </li>
</ul>

```

Figure 3.2: Sentences with labeled arguments in Rochester Data [5].

```

[
  {
    "rawText": "But their 1987 performance indicates that [ARGO they] [ARGM-MOD wo] [ARGM-NEG n't] [rel abandon] [ARG1 stocks] [ARGM-ADV unless conditions get far worse] .",
    "text": "But their 1987 performance indicates that they wo n't abandon stocks unless conditions get far worse .",
    "arguments": [
      {
        "type": "ARGO",
        "value": "they"
      },
      {
        "type": "ARGM-MOD",
        "value": "wo"
      },
      {
        "type": "ARGM-NEG",
        "value": "n't"
      },
      {
        "type": "rel",
        "value": "abandon"
      },
      {
        "type": "ARG1",
        "value": "stocks"
      },
      {
        "type": "ARGM-ADV",
        "value": "unless conditions get far worse"
      }
    ]
  },
  {
    "rawText": "And they believe 0 [ARGO the Big Board] , [ARGM-MNR under Mr. Phelan] , has [rel abandoned] [ARG1 their interest] .",
    "text": "And they believe 0 the Big Board , under Mr. Phelan , has abandoned their interest .",
    "arguments": [
      {
        "type": "ARGO",
        "value": "the Big Board"
      },
      {
        "type": "ARGM-MNR",
        "value": "under Mr. Phelan"
      },
      {
        "type": "rel",
        "value": "abandoned"
      },
      {
        "type": "ARG1",
        "value": "their interest"
      }
    ]
  }
]

```

Figure 3.3: Temporary data in JSON format for the parsed data.

After the parsing process is completed for each verb and sense, a text file is created under the target data folder named with the related verb and sense id. Each data

Table 3.1: Details of the parsed JSON data format.

<code>rawText</code>	Raw text found in the resource data
<code>text</code>	Parsed text which is purified from tags, etc.
<code>arguments</code>	JSON array that includes argument data
<code>arguments</code> \rightarrow <code>type</code>	Type of the argument like ARG0, ARG1, ARGM-MOD, etc.
<code>arguments</code> \rightarrow <code>value</code>	Word(s) which is labeled with argument type

file contains a JSON array, and each item in the array consists from the data for a sentence. In these items, the raw and parsed sentence data is stored. An example representation of the parsed data is given in Figure 3.3 and the details of the related data keys are given in Table 3.1.

Finally, parsed data is converted into the type of the Turkish PropBank data by another researcher for the comparisons and future usages.

3.2 Assignment of Predicate and Arguments in Turkish PropBank

As described in the previous chapters, previous works on other languages were realized using supervised machine learning algorithms. As with other studies, we have used supervised computer learning in the research. Therefore, we need the previously prepared PropBank dataset for Turkish language to train and test the system. To achieve this purpose and support similar NLP works for Turkish language, we started to prepare a proposition bank for Turkish [1]. For this purpose, we use an homemade developed NLP toolkit which makes possible to realize lots of useful NLP tasks for Turkish PropBank. In this thesis, only the predicate and argument editor parts of this tool is explained.

3.2.1 Selection of Predicates in Turkish PropBank Toolkit

First of all, we started with annotating the predicate of each sentences using *Propbank Predicate Editor* of this NLP toolkit. In Figure 3.4, a sample screenshot is given for the PropBank Predicate Editor. In this tool, it is possible to assign

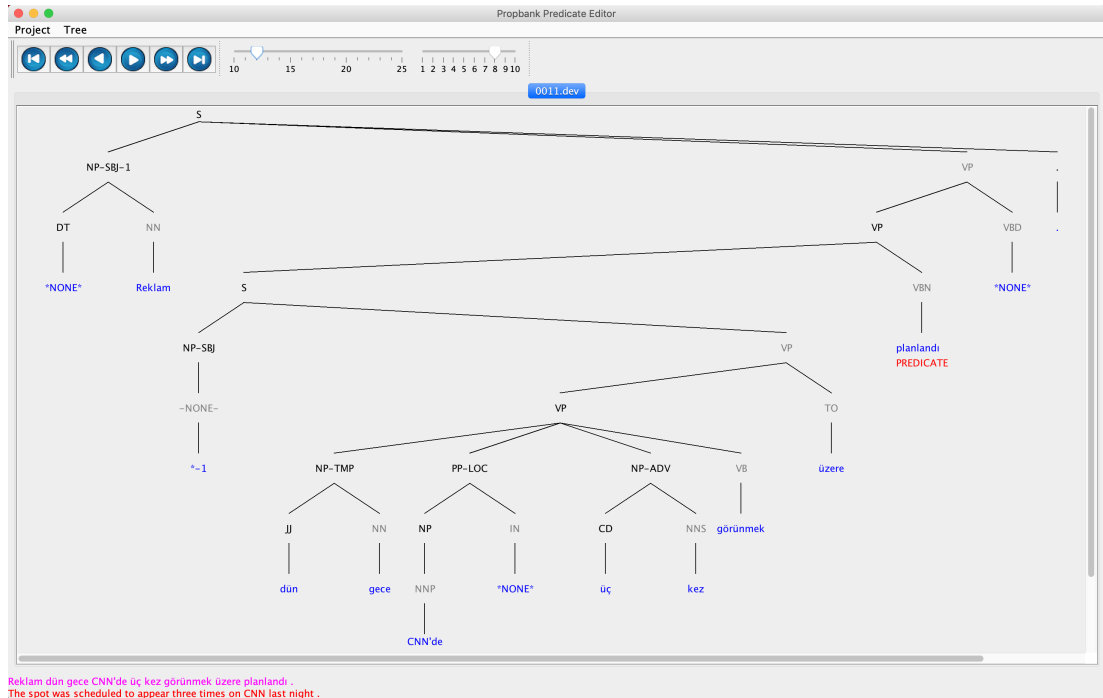


Figure 3.4: A screenshot of the PropBank Predicate Editor Tool.

predicate label to the related item by clicking over it and selecting as a predicate, as shown in Figure 3.5.

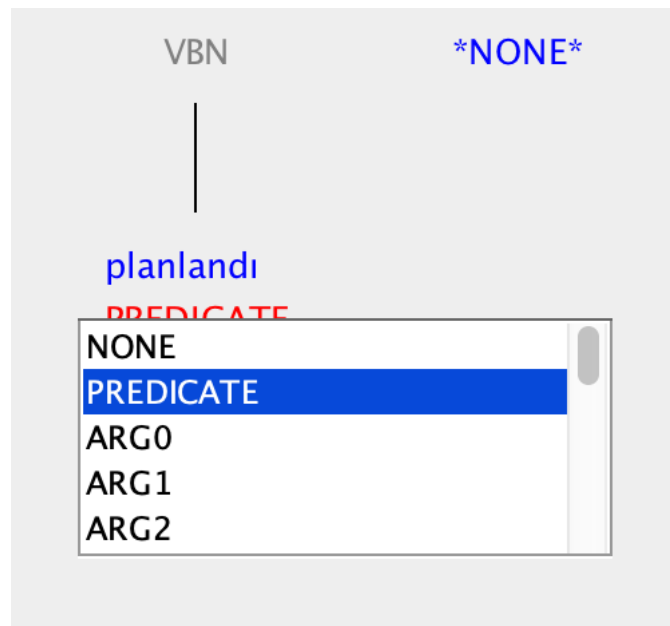


Figure 3.5: Selecting Predicates in PropBank Predicate Editor Tool.

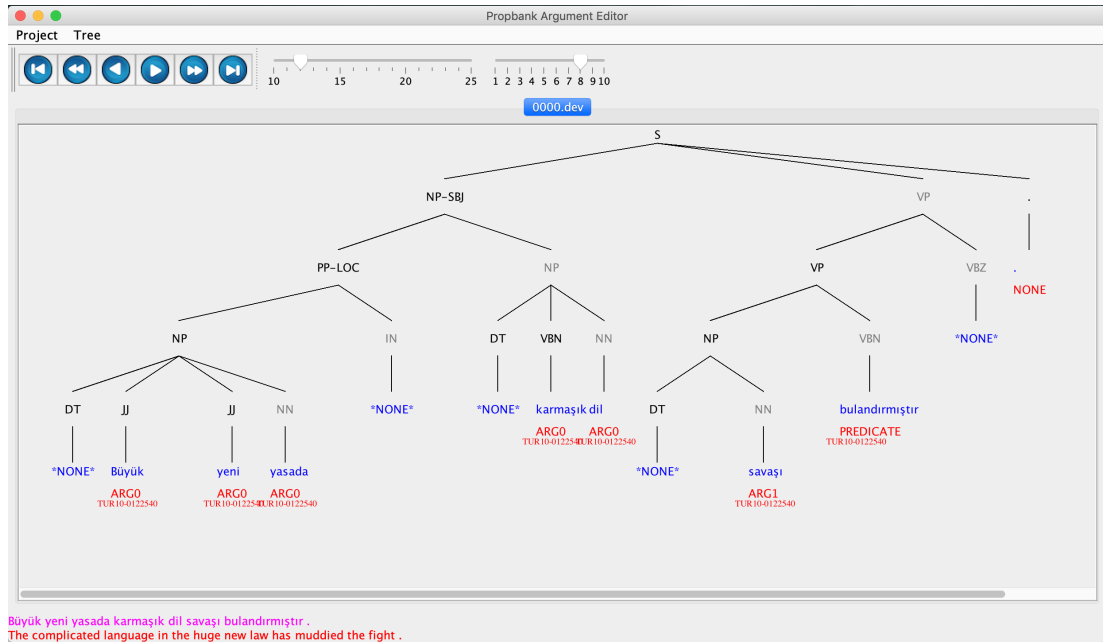


Figure 3.6: A sample screenshot for the PropBank Argument Editor Tool.

3.2.2 Selection of Arguments in Turkish PropBank Toolkit

After the selection of predicates found in sentences, selection of the arguments related with the each found predicate is completed using *PropBank Argument Editor* of the NLP Toolkit. A screenshot of the PropBank Argument Editor is presented in Figure 3.6. In this tool, when an item is selected, a popup window is opened with the possible argument list which allows to assign labels to the constituents. An example for the argument selection windows is given in Figure 3.7.

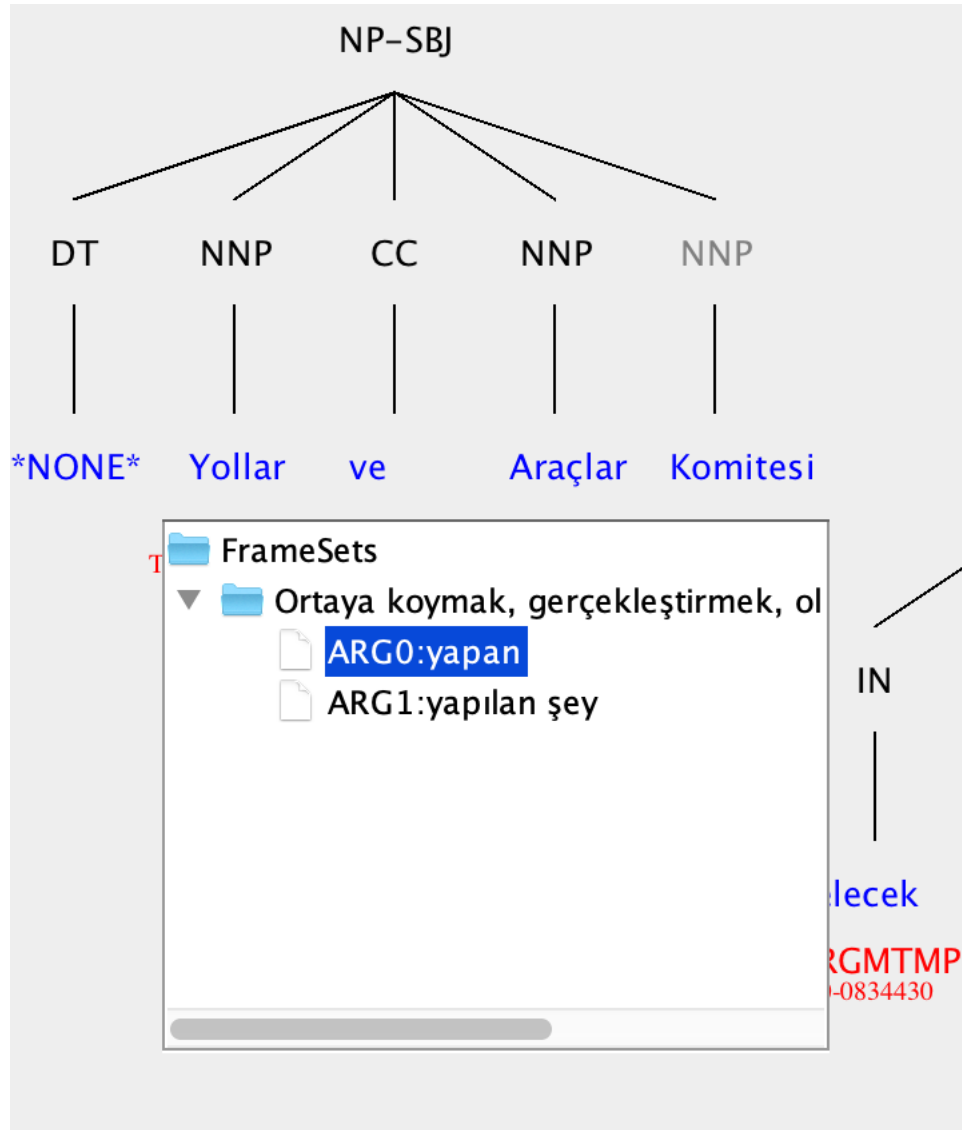


Figure 3.7: Selecting Arguments in PropBank Argument Editor Tool.

Chapter 4

PropBank Website

In the study, a website is prepared to present the PropBank data with the annotated sentences in public. PropBank website contains English sentences and their Turkish translations. All of the annotated predicates and their arguments are also presented with PropBank tree for both of the languages. Thus, it is so useful to see and compare PropBank for these languages.

PropBank website is publicly available [6] for the researchers who wants to work on Turkish PropBank data.

4.1 Preparation of the PropBank Website

In Turkish PropBank website [6], sentences have annotated predicates and arguments with their tree structured in both Turkish and English. Turkish data is generated with manually using the NLP toolking by human taggers, as written in Section 3.2. For the original sentences in English PropBank, all of the predicates and their arguments are parsed from the publicly accessible Rochester Data [5] as written in Section 3.1.

In the home page of the PropBank website, all of the defined Turkish predicates are listed line by line. The number located after each predicate and among the brackets shows the number of senses for the given predicate. A search box is also

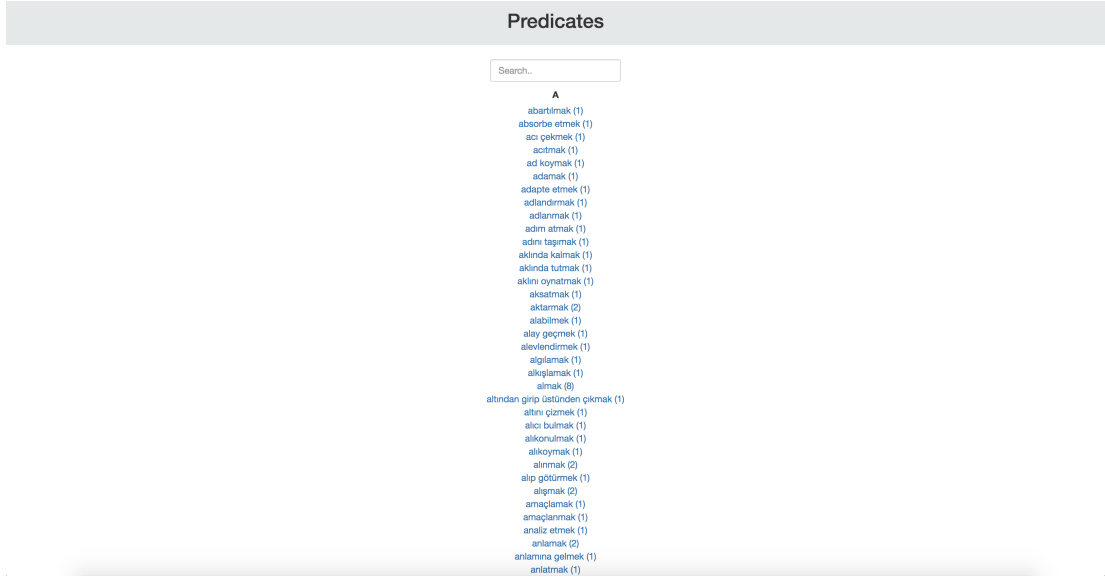


Figure 4.1: Homepage - List of Predicates found in Turkish PropBank.

available at the top of the page to filter the predicates matched with the entered text. A screenshot of the homepage with the predicates are given in Figure 4.1.

Each of the listed predicates has a link to the details page which shows the details of the selected predicate. At the details page, the name of the selected predicate is located at the top of the page with an orange background. In Figure 4.2, a screenshot is available for the predicate details page.

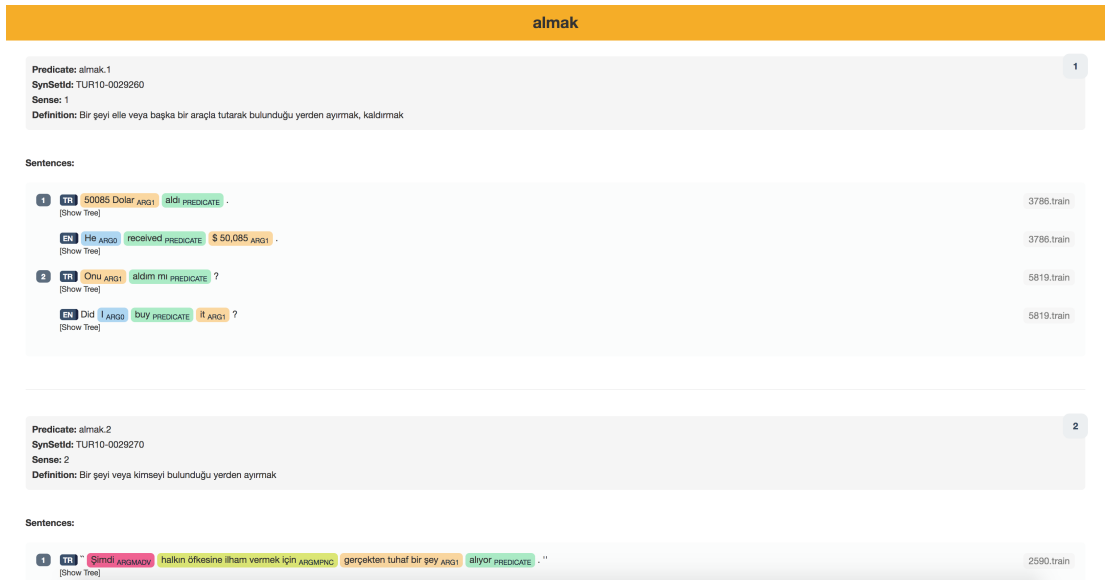


Figure 4.2: Page - Predicate Details.


















All of the available sense are available in the details page related with the predicate queried predicate and each of the sense is separated with a horizontal grey line. For each sense, there are two light grey boxes located one under the other.

The first box gives information about the predicate and current sense with the Turkish definition of it, and the sense id is presented at the right upper corner of the box with a darker grey background.

In the second box of the sense, all of the available sentences are given with both in Turkish and in English. Each sentences found in the box for the related sense id is numbered starting with 1 and each number is located at the left side of the sentences. Even if the sentences are shown in different languages, they are shown with the same number because they have the same meaning. An ISO two-letter language code shows the language of the sentence. The name of the sentence data file is also presented at the right side of each sentence within a darker grey box. In each sentence, each of the assigned predicates and their arguments are marked with a subtext like ARG0, ARG1, PREDICATE, etc., and also each type of argument has a background in a different color. Current schema of available colors for each argument is given in Table 4.1.

Using the “Show Tree” link found under the sentences, it is possible to show the tree structure with the semantic role labels of the relevant sentence for both of the languages, Turkish and English. When a tree view is opened with this link,

Table 4.1: Color schema for Semantic Role Labels.

ARG0		ARGMDIR	
ARG1		ARGMDIS	
ARG2		ARGMEXT	
ARG3		ARGMLOC	
ARG4		ARGMMNR	
ARG5		ARGMTMP	
ARGMADV		ARGMNONE	
ARGMCAU		ARGMPNC	
PREDICATE			

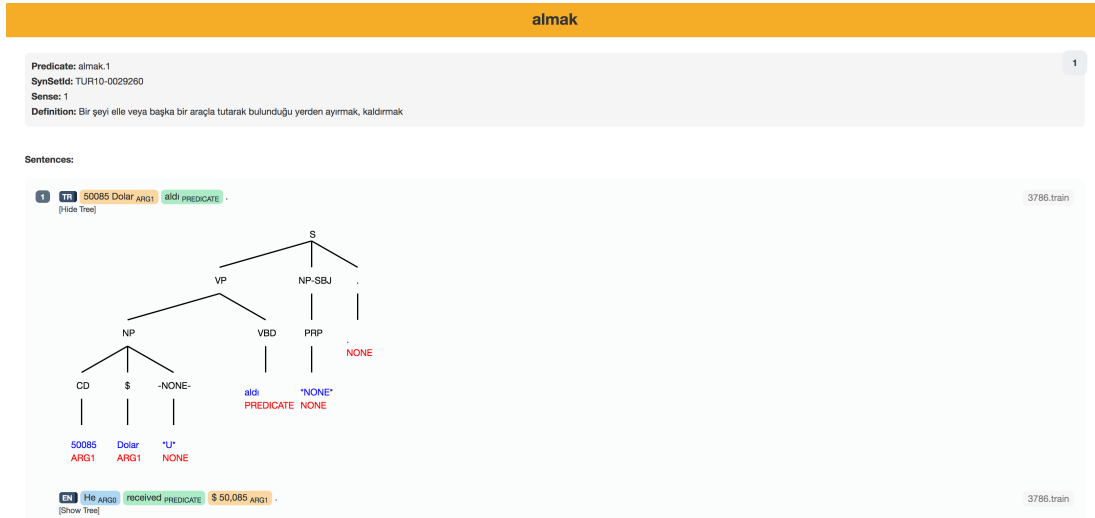


Figure 4.3: Page - Show Tree in Predicate Details.

the text will be “Hide Tree” which hides the open tree view. Figure 4.3 shows a tree view for the selected sentence.

4.2 Technologies Used in the Study

A Java application is developed using Spring Framework to prepare and provide data model on the backend side. Data model is prepared and provided via RESTful APIs and content type of the transferred data is JSON. The developed Java application works using a Tomcat Application Server.

A web application is developed using AngularJS Framework and CSS to present data model using a web browser. The AngularJS application calls the RESTful APIs to get data provided by the Java application and prepares the HTML view. CSS was used to get a good view on frontend side.

The frontend code is located under “src → main → resources → static” package path. Table 4.2 shows the paths for AngularJS application.

AngularJS codes are located under “static → js” package path. “app.js” file is the main AngularJS file that initializes the frontend application. This file also includes the route provider that routes requested path to the related controller.

Table 4.2: Web Application Paths for AngularJS.

Path	Explanation
static → css	CSS code files
static → js	Main AngularJS codes in Javascript
static → libs	Used libraries like AngularJS
static → views	HTML views for AngularJS

Chapter 5

Features

In this section, all of the feature used to assign semantic role labels are explained in two sections as “Word-Based Features” and “Tree-Based Features”.

5.1 Word-Based Features

5.1.1 Predicate

The predicate of the sentence. Note that all of the arguments are being classified according to this predicate.

Example: The lawyers [went PREDICATE] to work.

5.1.2 Phrase Type

Syntactic category (NP, PP, etc.) of the classified component.

Since different syntax categories tend to be different types of arguments, the phrase type of the classified component is a useful feature, but when such predicate is known, such trends are even stronger. A sample sentence with the phrase types of the components is given in Figure 5.1. The phrase type can be assigned to any number of argument tags in the SBAR itself, but the argument tag is more precise when the predicate is “ask”.

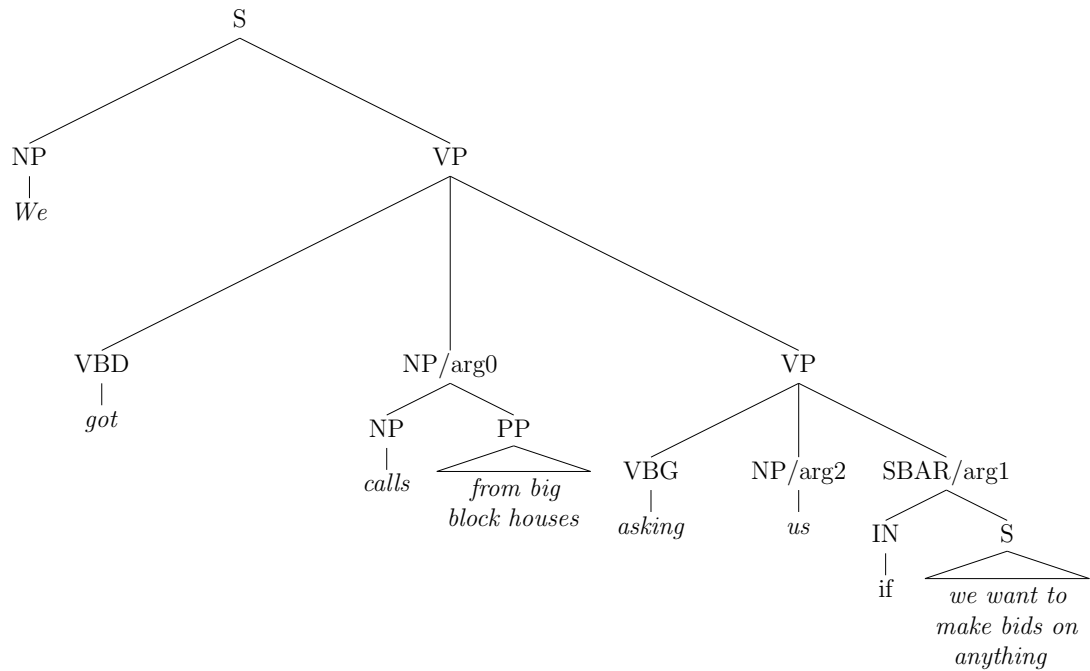


Figure 5.1: Lexicalization of phrase type features.

5.1.3 Voice

The recognition of active and passive verbs is very important to predict semantic roles because active verbs have a correlation of semantic role with the subjects of passive verbs. For example, a subject NP in an active sentence as ARG0 may be more probable to being ARG1 in a passive sentence.

Note that this feature is common in parse tree and it is shared by all of the components.

5.1.4 Named Entities in Constituents

There are infinite number of proper names for the named entities, especially for the people, organizations and locations. It is possible to identify them using named entity recognizer for the instances of classes PERSON, ORGANIZATION, LOCATION, PERCENT, MONEY, TIME, and DATE [7].

Note that named entity recognizer uses the entity labels as features to identify words found in a sentence.

5.2 Tree-Based Features

5.2.1 Path

The shortest path from categorized element to the predicate.

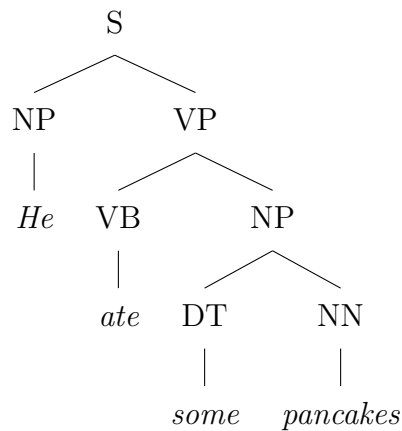


Figure 5.2: Sample tree for the path feature.

In the given sample Figure 5.2, the **predicate** of the sentence is “*ate*” and the **path** from predicate to the argument “**NN**” “*pancakes*” is VB \uparrow VP \downarrow NP \downarrow NN. Note that \uparrow indicates upward movement and \downarrow indicates downward movement in the parse tree.

According to several researchers (Palmer et al.) [4], *path* does not perform the argument classification job well. One of the reasons the path is ineffective is that children attached to a parse tree at the same level can not be distinguished. Despite all this, the path is very beneficial as it represents the relation between predicate and many sorts of grammatical functions.

5.2.2 Position

This feature specifies the relative location (before or after) of the classified component according to the predicate. It enables to overcome errors as a result of incorrect parsing. Gildea and Jurafsky [8] also offered position feature to see

how much can be accomplished without tree parsing. Subjects are usually found before verbs, but after objects. For this reason, the relation of this property to the grammatical function is expected to be high. An illustration of the position feature is given in Figure 5.3.

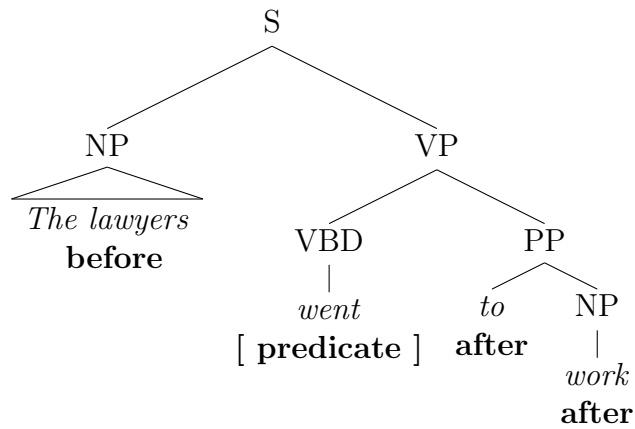


Figure 5.3: Illustration of the position feature.

5.2.3 Sub-categorization

Subcategorization feature is a phase-structure rule which expands the parent node of the predicate in the parse tree. For the given sentence in Figure 5.4 with the predicate *open* [8], the value of the feature is “VP → VB NP”. It is important to make it clear that this feature is also common in parse tree and it is shared by all of the components. For this reason, this feature does not differentiate between components in different syntactic positions.

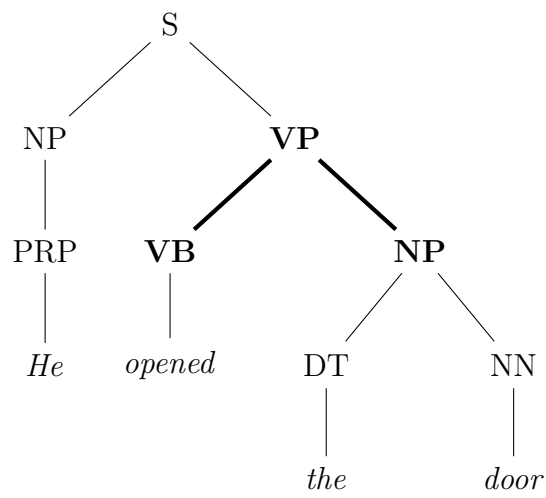


Figure 5.4: Subcategorization example for the predicate *open*.

Chapter 6

Classifiers

In this section, all of the classifiers used to assign semantic role labels are explained in general.

6.1 Rocchio

By using the centroids, Rocchio constitutes a simple representative for each class.

To compute the centroid of each class:

$$\vec{\mu}(c) = \frac{1}{|Dc|} \sum_{d \in Dc} \vec{v}(d) \quad (6.1)$$

where:

Dc : is the set of all documents that belong to class c

$v(d)$: is the vector space representation of d

In Rocchio classification [10] [9], the set of points with equal distance from the two distances is the boundary between two classes. To arrange a new item identify which centroid is nearest to and appoint it to the relating class c_i . For example, in Figure 6.1, $|a_1| = |a_2|$, $|b_1| = |b_2|$, and $|c_1| = |c_2|$.

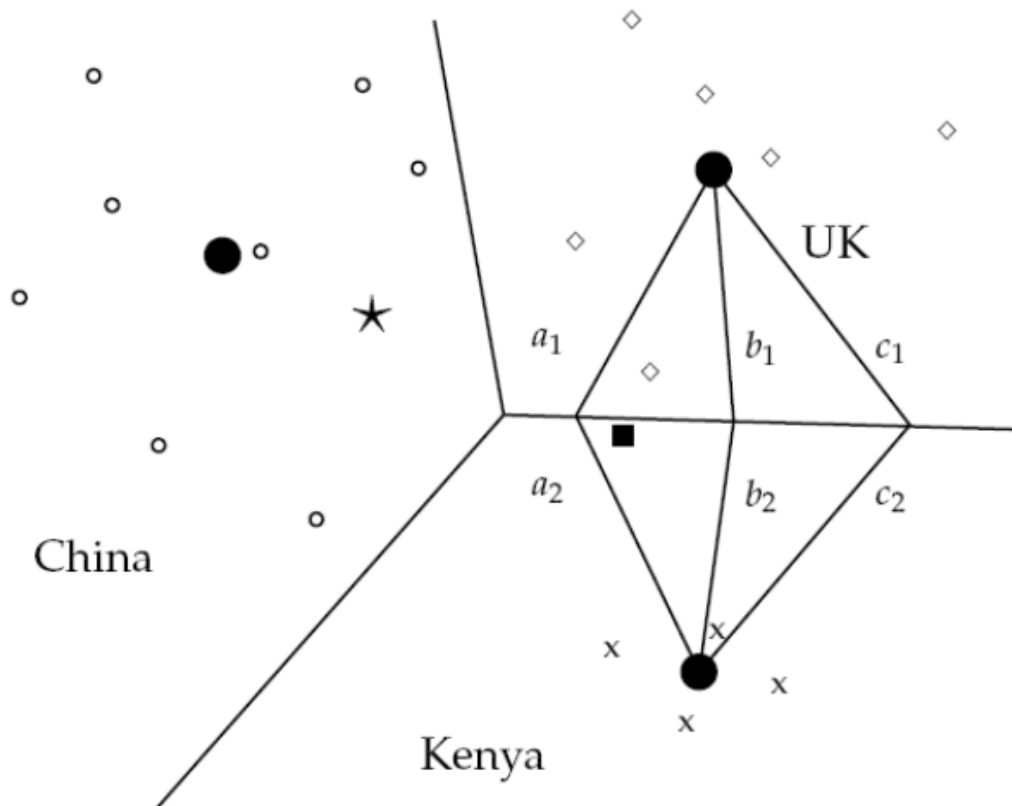


Figure 6.1: Rochhio Classification [9].

6.2 Naive Bayes

Naive Bayes is a predictive and descriptive classification algorithm that analyzes the relationship between the target variable and independent variables. Naive Bayes does not work with continuous data. Therefore, dependent or independent variables that containing continuous values should be made categorical. [11]

Naive Bayes is a probabilistic learning method based on Bayes rule which is given at Equation 6.2. [12]

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)} \quad (6.2)$$

where:

d : is a document

c : is a class

$$c_{map} = \operatorname{argmax}_{c \in C} P(c | d) = \operatorname{argmax}_{c \in C} P(c) \prod_{1 \leq k \leq n_d} P(t_k | c) \quad (6.3)$$

where:

map : maximum a posteriori (most likely class)

d : is a document

c : is a class

$P(t_k | c)$: is the conditional probability of term t_k occurring in a document of class c

$P(c)$: is the prior probability of a document occurring in class c

n_d : is the number of such tokens in d that are part of the vocabulary used for classification

The calculation of this formula according to probability values is sometimes complicated. For this purpose, it is recommended that the values to be used are taken as natural frequencies and not as a percentage of frequency.

When the Naive Bayes algorithm is used in the textual data, the textual data is first categorized manually and then the algorithm can learn from this training data and label the next unlabeled data. [11]

6.3 kNN

The kNN algorithm is a classification method. With this method, the sample data point class and the nearest neighbor class are determined according to the

k value. This method is the simple, older, most-known and effective pattern classification method. Because of these advantages, kNN algorithm is one of the most common machine learning algorithm. [13] [14] [15] [16] [17]

Object classification is a significant research topic. It is applied in data mining, statistics, artificial intelligence, cognitive psychology, medicine, pattern recognition and a wide range of fields. [13]

kNN algorithm is mostly used in classification applications because of the following advantages:

1. easy to apply,
2. does not require a training,
3. parallel implementation availability,
4. traceable,
5. noisy training set resistance,
6. easy to adapt to local data,

Even though the advantages, kNN algorithm has some disadvantages listed below:

1. requires a large amount of memory space,
2. the increase in processing load and cost as the data set and attribute size increase significantly,
3. performance is influenced by parameters and properties like distance measure, number of neighbor and attributes. [13] [14] [18]

6.4 Linear Perceptron

It is assumed that there are p of \vec{x} vectors which have real-valued numerical input features. In this case, Y is a binary class of the response. There are only two output categories as 1 and +1. A linear separator divides these two categories in the feature space. The classification can be defined as:

When $p = 2$, the space is separated with a line,

When $p = 3$, the space is separated with a plane,

Generally, the dimension of the hyper plane is $(p - 1)$ in a space with p -dimension.

A vector \vec{w} which is perpendicular to the orientation of a plane can be used to define the orientation of the plane.

There are infinite number of parallel planes have the same orientation. The closest plane to the origin is selected. The following formula can be expressed to define each the class of the points on each side of the plane:

$$\hat{y}(\vec{x}, b, \vec{w}) = \text{sgn } b + \vec{x} \cdot \vec{w} \quad (6.4)$$

where:

$$\vec{x} \cdot \vec{w} = \sum_{j=1}^p x_j w_j$$
$$\text{sgn } b = \begin{cases} +1 & \text{If its argument is } > 0 \\ 0 & \text{If its argument is } = 0 \\ -1 & \text{If its argument is } < 0 \end{cases}$$

\vec{w} is a unit vector and $\vec{x} \cdot \vec{w}$ is the projection of \vec{x} on that direction. [19]

6.5 Multi Layer Perceptron

Multi Layer Perceptrons are developed upon the failure of Single Layer Perceptrons to solve non-linear problems, consist of an input layer in which information is entered, one or more hidden layers and an output layer. The MLP has transitions between the layers, called forward and back propagation. In the forward propagation phase, the output and error value of the network is calculated. In the back propagation phase, the inter-layer link weight values are updated to minimize the calculated error value. [20]

Chapter 7

Experiments

7.1 Setup

There were a total of 9561 sentences in Turkish PropBank where the experiment was conducted. 3248 of the sentence files were excluded from the experiment since the predicates of these sentences are unknown. In experiments, only 6313 of 9561 files were considered because most of the features for semantic role labeling are related with the predicate of the sentence. Table 7.1 and Table 7.2 presents the details of predicates and class labels found in Turkish PropBank respectively.

Table 7.1: Setup Files with Detailed Predicate Information.

# of Total Files in Turkish PropBank	9561
# of Total Files with at least one Predicate Assigned	6313
# of Total Files with No Predicate Assigned	3248
# of Total Files with Multiple Predicates	307
# of All Predicates found in Files with Multiple Predicates	638
# of Files Considered in Experiment	6313

Table 7.2: Number of class labels in Turkish PropBank.

NONE	9305	ARG4	117	ARGMTMP	1776
ARG0	8094	ARGMPNC	702	ARGMADV	198
ARG1	13377	ARGMEXT	1210	ARGMMNR	1048
ARG2	851	ARGMLOC	898	ARGMDIR	58
ARG3	87	ARGMCAU	204	ARGMDIS	735

7.2 Prediction of the Predicates

Since the predicate must be known for the next stages, the experiment was started by trying to estimate the predicates. First of all, all of the leaf nodes in Turkish representation of the sentence were checked if the node has a “VERB” root POS or POS tag. When a match was found, the node was accepted as a predicate. In the basic comparison with the manually annotated data, the result was the like in Table 7.3 and accuracy was 80.9%. For the results of each stage, accuracy was calculated using the Equation 7.1.

$$Accuracy = \frac{CP}{AP + WP} * 100 \quad (7.1)$$

where:

CP : # of Correct Predictions

AP : # of All Predicates found in Files

WP : # of Wrong Predictions

Additionally, precision and recall values were calculated using the Equation 7.2.

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \quad (7.2)$$

where:

TP : True Positives

FP : False Positives

FN : False Negatives

Table 7.3: Result of first experiment to detect predicates.

# of All Files Considered	6313
# of Files with No Predictions	122
# of All Predicates found in Files	6644
# of Correct Predictions	5741
# of Wrong Predictions	450
Accuracy	80.9%
Precision	0.93
Recall	0.86

When the results were examined, there were some words which the root of the word is verb but it does not a predicate of the sentence. In Turkish language, the predicate is usually located in the last part of the sentence. In the example sentence “Görünüşe göre komisyon gerçekten bu ideale [inanmadı PREDICATE].”, although the predicate of the sentence is “inanmadı”, “Görünüşe göre” was determined as predicate in the result of the first experiment. The reason for this error was that the process was done by examining the morphological labels of the words. Note that the verbs in the roots of the words can be misunderstood. For the given example, the root of the “Görünüşe” word is “Görün” refers to the verb “Görünmek” in Turkish PropBank data.

In Turkish language, the predicate is usually located in the last part of the sentence and this was the reason why the predicate was determined wrongly. For this reason, all of the leaf nodes in Turkish representation of the sentences were reversed and the experiment was repeated. As can be seen from the results of the experiment in Table 7.4, the accuracy of the experiment was increased to 92.3%. When the results were examined, the number of correct predictions are increasing while the number of incorrect predictions are decreasing.

If there is more than one activity in a sentence, it may be possible to have more than one predicate. This was also the reason why there were 6644 predicates, although there were only 6313 sentences in the setup data. For this reason, even if a predicate is found, the examination process should continue. After this change was applied, the accuracy was increased to 95.7%, as shown in Table 7.5. When

Table 7.4: Result of second experiment to detect predicates : Reverse all of the leaf nodes.

# of All Files Considered	6313
# of Files with No Predictions	122
# of All Predicates found in Files	6644
# of Correct Predictions	6161
# of Wrong Predictions	29
Accuracy	92.3%
Precision	0.99
Recall	0.93

the presence of multiple predicates in a sentence was considered, the number of correct predictions was increased from 6161 to 6396, but only 8 increases seen in the number of incorrect predictions.

Table 7.5: Result of third experiment to detect predicates : Added examination of multiple predicates.

# of All Files Considered	6313
# of Files with No Predictions	122
# of All Predicates found in Files	6644
# of Correct Predictions	6396
# of Wrong Predictions	37
Accuracy	95.7%
Precision	0.99
Recall	0.96

7.3 Prediction of the Arguments

Different type of features were used in the experiment. Firstly, all features were developed in the toolkit to work on the Turkish language. In all phases of the experiment, all of the features were used to run different classifiers. All of the tests were executed using stratified k-Fold cross validation technique. Execution results with 3 different window sizes were presented in Table 7.6. Note that it is not meaningful to increase window size too much, because when the size of the window was increased, it becomes worse for the sentences with only a few words.

First of all, we started our experiments with using window size as 1 to generate instances. In this stage, the accuracy of the test was 36.1% for Naive Bayes

Table 7.6: Argument Prediction Results for different window sizes.

Classifier	Accuracy (for window size: n)		
	$n = 1$	$n = 2$	$n = 3$
Rocchio	49.2 \pm 1.7	31.0 \pm 1.4	39.1 \pm 2.7
Naive Bayes	36.1 \pm 1.3	31.1 \pm 0.6	27.9 \pm 1.8
kNN	65.0 \pm 0.7	65.7 \pm 1.0	68.8 \pm 0.9
Linear Perceptron	73.1 \pm 1.6	75.6 \pm 1.8	77.7 \pm2.4
Multi-Layer Perceptron	72.2 \pm 1.8	73.8 \pm 2.3	75.1 \pm 2.3

classifier, which is the worst result in this window size. Then, as we began to use different classifiers, we found that the results improved. The best result is 73.1% with Linear Perceptron classifier for window size 1.

In the second stage, we increased the window size to 2 and retried the experiment. In this stage, Rocchio and Naive Bayes classifiers gave the worst results. We gain the best result from Linear Perceptron classifier with 75.6% accuracy. The performance of the Multi-Layer Perceptron was very close to the result of Linear Perceptron.

In the third stage, the windows size of the instance generator was assigned to 3. When we retried the same experiment, the results were better and Linear Perceptron classifier performed best with 77.7% accuracy.

In general, Linear Perceptron was the best performing classifier among the classifiers we used. Performance of the Multi-Layer Perceptron and kNN classifiers were also close to the performance of Linear Perceptron. The results from the Rocchio and Naive Bayes classifiers were not satisfactory at all. Even if the results were not perfect, it was quite satisfactory.

For the window size 3, confusion matrices with average values are given in Table 7.7 and Table 7.8 for the best and worst performing classifiers.

An example representation of the generated feature data set is presented in Figure 7.1.

bulandırmıştır;VP -> VP VBZ;DT↑ NP↑ NP-SBJ↑ S↓ VP↓ VP↓ VBN↓
;NP;BEFORE;ACTIVE;NONE;PREDICATE

çalar;S -> NP-SBJ VP .;NNP↑ NP↑ VP↓ VBZ↓
;NP;BEFORE;ACTIVE;NONE;PREDICATE

başlamıştı;VP -> ADVP-TMP VP VBZ;DT↑ NP-SBJ↑ S↓ VP↓ VP↓ VBN↓
;NP-SBJ;BEFORE;ACTIVE;NONE;ARGMTMP

başlamıştı;VP -> ADVP-TMP VP VBZ;NN↑ NP-SBJ↑ S↓ VP↓ VP↓ VBN↓
;NP-SBJ;BEFORE;ACTIVE;NONE;PREDICATE

başlamıştı;VP -> ADVP-TMP VP VBZ;RB↑ ADVP-TMP↑ VP↓ VP↓ VBN↓
;ADVP-TMP;BEFORE;ACTIVE;NONE;NONE

sattı;S -> NP-SBJ NP-TMP VP;NNP↑ NP↑ PP-LOC↑ VP↓ VBD↓
;NP;BEFORE;ACTIVE;LOCATION;PREDICATE

artırdı;S -> NP-SBJ VP .;NNP↑ NP-SBJ↑ S↓ VP↓ VBD↓
;NP-SBJ;BEFORE;ACTIVE;ORGANIZATION;ARGO

artırdı;S -> NP-SBJ VP .;NNP↑ NP-SBJ↑ S↓ VP↓ VBD↓
;NP-SBJ;BEFORE;ACTIVE;ORGANIZATION;ARGO

artırdı;S -> NP-SBJ VP .;NNP↑ NP-SBJ↑ S↓ VP↓ VBD↓
;NP-SBJ;BEFORE;ACTIVE;ORGANIZATION;ARG1

Figure 7.1: Sample feature data extracted from Turkish PropBank.

Table 7.7: Confusion Matrix for the Naive Bayes classifier executed with window size 3.

	NONE	ARG0	ARG1	ARG2	ARG3	ARG4	ARGMPNC	ARGMEXT	ARGMLOC	ARGMCAU	ARGMTMP	ARGMADV	ARGMMNR	ARGMDIR	ARGMDIS
NONE	11.0	0.4	0.5	0.3	0.6	0.1	1.3	1.7	0.3	0.2	1.1	0.2	0.5	0.6	0.6
ARG0	1.6	21.1	1.6	0.1	1.9	0.0	2.0	3.7	0.7	3.0	4.5	3.1	3.1	1.4	4.9
ARG1	5.1	21.0	25.2	1.8	12.2	0.4	8.6	16.2	6.1	14.3	25.6	21.6	18.7	8.5	37.3
ARG2	0.0	0.0	0.0	1.1	0.8	0.0	0.1	0.0	0.0	0.0	0.3	1.1	0.0	0.0	0.2
ARG3	0.3	0.0	0.2	0.1	6.6	0.0	0.3	0.9	0.1	0.0	0.3	0.1	0.2	0.2	0.0
ARG4	0.8	0.0	0.0	0.0	0.1	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.0	1.6	0.1
ARGMPNC	0.4	0.2	0.1	0.1	0.7	0.1	5.6	0.5	0.3	0.2	0.2	0.5	0.4	0.2	0.3
ARGMEXT	1.0	0.5	0.3	0.3	10.8	0.1	0.9	10.8	0.9	0.6	1.4	0.6	0.6	1.7	0.8
ARGMLOC	0.0	0.5	0.2	0.0	0.0	0.0	0.2	0.3	1.5	0.0	0.4	0.1	0.0	0.2	0.3
ARGMCAU	0.2	0.2	0.0	0.0	0.1	0.0	0.0	0.2	0.1	1.2	0.1	0.3	0.1	0.0	0.2
ARGMTMP	0.1	0.3	0.2	0.0	0.4	0.1	0.0	0.9	0.6	0.5	6.0	0.7	0.4	0.6	1.0
ARGMADV	0.0	0.4	0.2	0.2	0.3	0.0	0.3	0.7	0.0	0.2	1.2	12.5	0.9	1.4	0.9
ARGMMNR	0.6	0.1	0.1	0.1	0.2	0.1	0.3	1.3	0.2	0.9	0.5	1.0	6.4	0.7	1.1
ARGMDIR	0.3	0.7	0.3	0.0	0.2	0.1	0.6	1.0	0.2	0.5	0.2	0.8	0.4	13.8	0.7
ARGMDIS	2.0	4.0	5.5	0.2	1.6	0.0	4.0	5.6	0.8	0.0	13.2	1.8	2.3	3.1	9.6

Table 7.8: Confusion Matrix for the Linear Perceptron classifier executed with window size 3.

	NONE	ARG0	ARG1	ARG2	ARG3	ARG4	ARGMPNC	ARGMEXT	ARGMLOC	ARGMCAU	ARGMTMP	ARGMADV	ARGMMNR	ARGMDIR	ARGMDIS
NONE	8.3	0.4	2.7	0.1	0.2	0.1	0.1	0.2	0.0	0.1	0.1	0.1	0.3	0.2	2.9
ARG0	0.1	24.0	13.1	0.0	0.0	0.0	0.2	0.3	0.1	0.0	0.2	0.1	0.1	0.0	3.2
ARG1	0.2	4.2	172.7	0.1	0.2	0.0	0.3	0.8	0.0	0.1	0.8	1.9	0.2	1.2	1.0
ARG2	0.0	0.1	0.6	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1
ARG3	0.0	0.3	0.7	0.1	5.3	0.0	0.1	0.0	0.0	0.0	0.2	0.2	0.0	0.0	1.3
ARG4	0.1	0.1	0.1	0.0	0.1	2.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0
ARGMPNC	0.1	0.2	2.1	0.0	0.2	0.0	4.3	0.0	0.0	0.0	0.0	0.1	0.1	0.0	1.1
ARGMEXT	0.2	1.0	4.4	0.0	0.4	0.0	0.0	8.9	0.0	0.0	0.3	0.2	0.0	0.6	1.3
ARGMLOC	0.0	0.0	1.1	0.0	0.0	0.1	0.1	0.3	1.1	0.0	0.0	0.0	0.1	0.0	0.1
ARGMCAU	0.1	0.3	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.2	0.0	0.1	0.0
ARGMTMP	0.0	0.2	3.9	0.0	0.0	0.0	0.0	0.4	0.1	0.3	3.1	0.5	0.0	0.0	0.5
ARGMADV	0.3	0.5	3.5	0.1	0.1	0.0	0.0	0.2	0.0	0.0	0.1	9.7	0.1	0.6	1.0
ARGMMNR	0.3	0.1	3.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	5.6	0.2	1.0
ARGMDIR	0.3	0.5	2.8	0.0	0.0	0.2	0.0	0.3	0.0	0.0	0.0	0.5	0.0	10.4	0.6
ARGMDIS	0.1	1.7	2.7	0.0	0.6	0.0	0.3	0.6	0.0	0.0	0.0	0.2	0.0	0.4	43.5

Chapter 8

Conclusion

In this study, we contributed to the generation of Turkish PropBank. First of all, we started to annotate Turkish translation of the original PropBank sentences for semantic role labeling. Annotation of the PropBank data using manpower was a time-consuming process in general.

Then, semantic role tags in the original language were parsed to add them into Turkish PropBank. This process provided us to compare works and linguistic differences of Turkish and English. Additionally, a PropBank website was developed and made publicly available. Researchers working on Turkish PropBank started to use this website.

Finally, we studied on some semantic role labeling features to make process automatized. Linguistics structure of the Turkish language was also considered while implementing these features in NLP toolkit. Afterward, these features were used to train and test a system.

When the results are analyzed, it can be said that errors in the manually marked data have negative effects on the results. In the future, it may be possible to get better results by correcting errors in this data. The use of different classifiers and the addition of new features can further improve the results.

References

- [1] K. Ak, O. T. Yıldız, V. Esgel, and C. Toprak, “Construction of a Turkish proposition bank.” *Turkish Journal of Electrical Engineering and Computer Science*, no. 26, pp. 570–581, 2018.
- [2] M. A. Marcinkiewicz, M. P. Marcus, and B. Santorini, “Building a Large Annotated Corpus of English: The Penn Treebank.” *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
- [3] B. Santorini, “Part-of-Speech Tagging Guidelines for the Penn Treebank Project.” Technical Report, MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania.
- [4] N. Xue and M. Palmer, “Calibrating Features for Semantic Role Labeling.” pp. 88–94, 2004.
- [5] “Propbank Index in English.” <https://www.cs.rochester.edu/~gildea/PropBank/>, accessed: 2016-12-19.
- [6] “Propbank Index in Turkish.” <http://haydut.isikun.edu.tr/propbank-web/>, accessed: 2016-12-19.
- [7] M. Palmer, D. Gildea, and N. Xue, *Semantic Role Labeling*. Morgan & Claypool Publishers, pp. 39–42, 2011.
- [8] D. Gildea and D. Jurafsky, “Automatic Labeling of Semantic Roles.” *Computational Linguistics*, vol. 28, no. 3, pp. 245–288, 2002.
- [9] “Rocchio classification.” <https://nlp.stanford.edu/IR-book/html/htmledition/rochio-classification-1.html>, accessed: 2019-08-14.

- [10] “Text classification.” <https://cs.uwaterloo.ca/~mli/TextClassification.pdf>, accessed: 2019-08-14.
- [11] A. C. Gülce, “Veri Madenciliğinde Apriori Algoritması ve Apriori Algoritmasının Farklı Veri Kümelerinde Uygulanması.” 2010.
- [12] D. Jurafsky, “Text Classification and Naïve Bayes.” Lecture slides, December 2015.
- [13] E. Taşçı and A. Onan, “K-En Yakın Komşu Algoritması Parametrelerinin Sınıflandırma Performansı Üzerine Etkisinin İncelenmesi.” *Akademik Bilişim*, 2016.
- [14] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification.” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [15] N. Bhatia and Vandana, “Survey of nearest neighbor techniques.” *International Journal of Computer Science and Information Security*, vol. 8, no. 2, pp. 302–305, 2010.
- [16] X. Y. Qiu, K. Kang, and H. X. Zhang, “Selection of kernel parameters for k-nn.” *IEEE International Joint Conference on Neural Networks*, pp. 61–65, 2008.
- [17] G. E. A. P. A. Batista and D. F. Silva, “How k-nearest neighbor parameters affect its performance.” *Simposio Argentino de Inteligencia Artificial*, pp. 95–106, 2009.
- [18] H. Liu and S. Zhang, “Noisy data elimination using mutual k-nearest neighbor for classification mining.” *Journal of Systems and Software*, vol. 85, no. 5, pp. 1067–1074, 2012.
- [19] C. Shalizi, “Linear classifiers and the perceptron algorithm.” November 2009.
- [20] A. Arı and M. E. Berberler, “Yapay sinir ağları ile tahmin ve sınıflandırma problemlerinin çözümü için arayüz tasarımı.” 2017.