

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**AUTOMATIZED TEST GENERATION AND
EVALUATION TOOL FROM LECTURE NOTES
FOR HIGH SCHOOL STUDENTS**

by

Önder Can SARI

March, 2019

İZMİR

AUTOMATIZED TEST GENERATION AND EVALUATION TOOL FROM LECTURE NOTES FOR HIGH SCHOOL STUDENTS

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of
Science in Computer Engineering**

by


Önder Can SARI

March, 2019

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**AUTOMATIZED TEST GENERATION AND EVALUATION TOOL FROM LECTURE NOTES FOR HIGH SCHOOL STUDENTS**” completed by **ÖNDER CAN SARI** under supervision of **ASST. PROF. DR. ÖZLEM AKTAŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Asst. Prof. Dr. Özlem AKTAŞ

Supervisor



Prof. Dr. Yalçın Gebi

(Jury Member)



Doç. Dr. Deniz Kılınç

(Jury Member)



Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. Prof. Dr. Özlem AKTAŞ for her guidance, support and encouragement throughout the development of this project.

I would also thank Prof. Dr. Yalçın ÇEBİ who introduced natural language processing (NLP) field to me, and Dr. Emel ALKİM for her precious help at the earlier stages of the project.

We had started working on NLP tasks for Turkish in our undergraduate education with a group of three, so I would like to thank my friends Caner ALTUNTAŞ and Ahmet Erdem KAHVECİ for their valuable ideas and contributions to this project.

This project is supported by Dokuz Eylül University Department of Scientific Research Projects (DEÜBAP), numbered as 2018.KB.FEN.015.

I have special thanks to my family for their endless support all along.

Önder Can SARI

AUTOMATIZED TEST GENERATION AND EVALUATION TOOL FROM LECTURE NOTES FOR HIGH SCHOOL STUDENTS

ABSTRACT

Education systems force the students to deal with numerous exams. It can be really difficult to keep motivation up while handling those exams. In today's world, information technologies have spread over to all levels of business sectors and disciplines, including education. Therefore, to make studying more target-driven and catchy, it is a good approach to use information technologies on this issue.

Within this thesis, a software project, which includes the processing of text-based Turkish lecture notes of secondary education students in history and geography domains and automatic test generation, is developed. Main objective here is to provide a computer mediated self-study opportunity by simplifying an examination process with isolating students from the question preparation burden. Three question types are enabled for selection, which are true-false, fill in the blanks and multiple choice.

Proposed system allows users to define constraints like number of questions and included question types before an exam generation, save generated tests or specific questions for further usage, save their exam results to keep track of their progress.

Besides the educational software developed, this research draws conclusions and proposes solutions on some of the major natural language processing (NLP) tasks for Turkish like document classification, detection of sentence boundaries and headings, conversion of verbs based on their polarities after classifying them as positive and negative, constructing glossary of terms structures for history and geography domains using named entity recognition (NER) techniques.

Keywords: Educational software, automatic test generation, natural language processing, Turkish, named entity recognition, glossary of terms, verb polarity conversion, sentence boundary detection, heading detection, document classification

LİSE ÖĞRENCİLERİ İÇİN DERS NOTLARINDAN OTOMATİK TEST OLUŞTURMA VE DEĞERLENDİRME ARACI

ÖZ

Öğrenciler eğitimleri boyunca çok sayıda sınava tabi tutulmaktadır. Bu sınavlarda başarılı olmaya çalışan öğrencilerin motivasyonunu yüksek tutması gerçekten zorlayıcı olabilir. Bugünün dünyasında, tüm iş kollarında ve bilim dallarında bilgi teknolojilerinden faydalanılmaktadır. Eğitim alanında da, süreci daha hedefe yönelik ve ilgi çekici hale getirmek için teknolojiyen yararlanmak güzel bir yaklaşımdır.

Bu tez kapsamında, ortaöğretim tarih ve coğrafya alanındaki metin içerikli Türkçe ders metinlerinin otomatik sınav oluşturmak için işlendiği bir yazılım projesi geliştirilmiştir. Ana hedef, öğrencileri soru hazırlama yükünden kurtararak test sürecini kolaylaştırmak ve onlara bilgisayar ortamında kendi kendilerini sınama imkânı sağlamaktır. Doğru-yanlış, boşluk doldurma ve çoktan seçmeli, kullanıcıların seçebileceği üç soru tipi olarak sistemde tanımlanmıştır.

Önerilen sistem kullanıcılara soru sayısı, dahil edilecek soru tipleri gibi kriterleri sınav oluşturmadan önce belirleme, oluşturulan sınavları ya da seçilen belirli soruları ileride kullanmak üzere kaydetme, sınav sonuçlarını kaydetme gibi olanaklar sağlar.

Geliştirilen eğitim yazılımının yanı sıra, araştırma Türkçe için bazı önemli doğal dil işleme (DDL) görevlerine yönelik sonuçlar ortaya koymuş ve çözüm önerileri geliştirmiştir. Bu görevler metin sınıflandırma, cümle sonu ve başlık belirleme, fiillerin olumlu ve olumsuz şeklinde sınıflandırılıp olumluların olumsuza, olumsuzların olumluya çevrilmesi, varlık ismi tanıma (VİT) teknikleri kullanarak tarih ve coğrafya alanları için terimler sözlüğü yapıları oluşturmaktır.

Anahtar kelimeler: Eğitim yazılımı, otomatik test oluşturma, doğal dil işleme, Türkçe, varlık ismi tanıma, terimler sözlüğü, fiillerde olumluluk - olumsuzluk dönüşümü, cümle sonu belirleme, başlık belirleme, metin sınıflandırma

CONTENTS

	Page
M.Sc THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF FIGURES	xii
LIST OF TABLES	xv
 CHAPTER ONE – INTRODUCTION	 1
1.1 Brief Description and Goals of Thesis	1
1.2 Brief Overview of NLP and Turkish Language	1
1.3 Task Distribution of Thesis	3
1.4 Development Environment of Thesis	3
1.5 Organization of Thesis	4
 CHAPTER TWO – TASK DEFINITION	 5
2.1 Document Classification	5
2.2 Detection of Sentence Boundaries and Headings	6
2.3 Detection and Conversion of Verb Polarity	7
2.4 Named Entity Recognition to Detect Terms	9
2.5 Automatized Question and Test Generation	11
 CHAPTER THREE – PREVIOUS WORK	 12
3.1 Document Classification	12
3.2 Sentence Boundary Detection	17
3.3 Stemming	22
3.4 Named Entity Recognition	29

3.5 Exam and Question Generation	35
CHAPTER FOUR – COURSE DOCUMENT CLASSIFICATION	39
4.1 Overview and Dataset Introduction	39
4.2 Compared Approaches	39
4.2.1 Existence of Stop Words	39
4.2.2 Stemming Approaches	40
4.2.3 Classification Algorithms	40
4.2.4 Feature Selection Methods	40
4.3 Experimentation Phase	40
4.3.1 Document Preprocessing	41
4.3.2 ARFF File Generation	42
4.3.3 Interpretation of Experiment Results	43
4.3.4 Selected Classification Model	47
4.4 Classification Model Integration with Exam Module	49
CHAPTER FIVE – SENTENCE BOUNDARY AND HEADING DETECTION.....	51
5.1 Overview	51
5.2 Regular Expression Usage	52
5.3 Software Structure of the Model	53
5.3.1 Initial Operations	54
5.3.2 Irrelevant Text Controls	54
5.3.3 Heading Format Controls	54
5.3.4 Paragraph Format Controls	55
5.3.5 Colon Character Controls	55
5.3.6 Head Character Controls	56
5.3.7 Operations to Join Itemized Text Parts	57
5.3.8 Text Normalization	59
5.3.9 Generation of Heading and Paragraph Objects	59

5.3.10 Sentence Boundary Detection	60
5.3.10.1 Division of Paragraphs	60
5.3.10.2 Abbreviation Controls	61
5.3.10.3 Quote Controls	61
5.3.10.4 Generation of Sentence Objects	62
5.4 Experimentation Phase	62
5.4.1 Used Dataset	62
5.4.2 Sentence Boundary and Heading Detection	63
5.4.3 Observation of Itemized Text Part Cases	65
 CHAPTER SIX – VERB POLARITY DETECTION AND CONVERSION	66
6.1 Overview	66
6.2 Classification Phase	67
6.2.1 Finite State Machine Structure	67
6.2.2 Base Class Definitions	70
6.2.2.1 State	70
6.2.2.2 Suffix	71
6.2.2.3 Transition	72
6.2.2.4 SuffixInWord	72
6.2.2.5 Path	73
6.2.2.6 FSM	74
6.2.3 Parse Operations	74
6.2.4 Path Elimination Operations	75
6.2.4.1 Elimination of Irrelevant Results	75
6.2.4.2 Elimination of Meaningless Stems	75
6.2.4.3 Elimination by Root Type	76
6.2.4.4 Elimination by Supervised Learning	76
6.3 Conversion Phase	77
6.3.1 Controls for Phonetic Rules	77
6.3.2 Suffix Formats	80
6.3.3 Conversion from Affirmative to Negative	82

6.3.4 Conversion from Negative to Affirmative	82
6.3.5 Optimal Result Decision	84
6.4 Use Case Example	84
6.5 Experimental Results	86

CHAPTER SEVEN – NAMED ENTITY RECOGNITION MODEL TO BUILD GLOSSARY OF TERMS STRUCTURE 89

7.1 Overview	89
7.2 Tokenizer and Tokens	90
7.3 Generated Sources for Lookup Operations	92
7.3.1 Lexical Model Sources	92
7.3.1.1 Final Exclusions from Lexical Sources	93
7.3.2 Contextual Model Sources	94
7.4 Labeling by Lexical and Contextual Models	95
7.5 Named Entities and Recognizer Model	97
7.6 Building Glossary of Terms Structure	99
7.6.1 NER Execution on Complete Dataset	99
7.6.2 Fine Grained Categorization	100
7.6.3 Database Model	102
7.6.3.1 NamedEntityType Table	103
7.6.3.2 Category Table	104
7.6.3.3 Term Table	105
7.6.3.4 SynonymTerm Table	106
7.7 Experimental Results	107
7.8 Encountered Challenges	111

CHAPTER EIGHT – AUTOMATIZED QUESTION AND TEST GENERATION 113

8.1 Overview	113
8.2 Automatized Question Generation	113

8.2.1 Contribution of SBD Model	113
8.2.2 Contribution of Document Classification Model	114
8.2.3 Using Verb Polarity for Question Generation	114
8.2.4 Using Glossary of Terms for Question Generation	115
8.2.5 Question Type Decision	117
8.3 Test Generation	119
8.3.1 Specifiable Criteria	119
8.3.2 Test Generation on New Document	120
8.3.3 Test Generation on Existing Document	120
8.4 Other Features of iTest	121
8.5 Database Model	126
8.5.1 SystemUser Table	127
8.5.2 UserLogin Table	128
8.5.3 Document Table	128
8.5.4 Exam Table	129
8.5.5 Question Table	130
8.5.6 FitbAnswer Table	131
8.5.7 ExamQuestion Table	132
8.5.8 ExamResult Table	132
CHAPTER NINE – CONCLUSION	133
9.1 Results and Evaluation	133
9.2 Future Enhancement	135
REFERENCES	136
APPENDICES	148
APPENDIX-1: Turkish Stop Words	148
APPENDIX-2: Most Distinctive 100 Words for Classification	150
APPENDIX-3: Turkish Abbreviation List	152

APPENDIX-4: Exceptional Verb Roots Affected by Consonant Lenition	153
APPENDIX-5: Example Lecture Note and Generated Test	154



LIST OF FIGURES

	Page
Figure 1.1 An example Turkish word parsed into its morphemes	2
Figure 2.1 Example document classification setup with six classes, each with three training documents and a test set document	5
Figure 2.2 A diagram showing relationships between the tasks defined within the thesis.....	11
Figure 3.1 Distribution of news text on first two dimensions	15
Figure 3.2 Dependency links and POS tags on an example sentence	25
Figure 3.3 Ontology used in experiments and classification results (detected hypernyms) for 7-unknown concepts	30
Figure 3.4 Information sources used for rule-based NER system	32
Figure 3.5 The exam testing page with an example generated question	36
Figure 4.1 Example output on different preprocessing stages of a raw document text.....	41
Figure 4.2 Standardized document example with No Stop Words and No Stemming approach	42
Figure 4.3 Standardized document example with No Stop Words and F5 stemming approach	42
Figure 4.4 ARFF file example for F6 stemming and no stop word removal approaches.....	43
Figure 4.5 Result screen of NB-B, F5, IG, NSW experiment (10% training – 90% test).....	48
Figure 4.6 Result screen of NB-M, ZS, CC, NSW experiment (50% training – 50% test)	48
Figure 5.1 Flow diagram of the proposed sentence boundary and heading detection model	53
Figure 5.2 Detection of headings with ordinal numbers	55
Figure 5.3 Detection of sub-headings at the beginning of paragraphs	56
Figure 5.4 Detection of sub-heading formatted itemized text parts	57
Figure 5.5 Example itemized text parts join operation output	58

Figure 5.6	Representation of how paragraphs are linked with headings	60
Figure 5.7	Example output when a sentence boundary is detected between a dot and an apostrophe	60
Figure 5.8	Output when a not-sentence boundary case is detected between a dot and an apostrophe	61
Figure 5.9	Example output when inner-sentence quote controls are disabled	62
Figure 5.10	Example output when inner-sentence quote controls are enabled	62
Figure 6.1	Sample FSM execution on a word with a proper predicate format	69
Figure 6.2	Example system output for verb polarity classification and stem detection tasks	76
Figure 6.3	Example system output for verb polarity classification and conversion tasks.....	86
Figure 6.4	Example case where multiple results remain after FSM execution and path elimination	88
Figure 7.1	Flow diagram of the proposed named entity recognition model	90
Figure 7.2	Example system output after tokenization and token labeling on an input sentence	97
Figure 7.3	Example system output that shows named entity detection on an input sentence	98
Figure 7.4	Diagram of the GlossaryOfTerms database	103
Figure 7.5	Data stored in NamedEntityType table	104
Figure 7.6	Sample data stored in Category table	105
Figure 7.7	Sample data stored in Term table	106
Figure 7.8	Sample data stored in SynonymTerm table	107
Figure 8.1	Example verb polarity detection and conversion output on a complete document	115
Figure 8.2	Example GoT lookup output to detect terms and their siblings within a sentence	116
Figure 8.3	Three types of questions generated from the same sentence	117
Figure 8.4	Generate test using new input document screen	120
Figure 8.5	Generate test using existing document screen	121
Figure 8.6	Register screen	122

Figure 8.7 Login screen	122
Figure 8.8 Examination screen	123
Figure 8.9 After examination screen	124
Figure 8.10 Find test screen with no filters are applied	124
Figure 8.11 Exam results screen when filter and sort operations are applied	125
Figure 8.12 Preferences screen	126
Figure 8.13 Diagram of the ITEST2018 database used for examination model operations	127
Figure 8.14 Sample data stored in Exam table	129
Figure 8.15 Example case where a synonym term is accepted as the correct answer.	131



LIST OF TABLES

	Page
Table 2.1 Changes on morpheme sequences based on the polarity condition of a predicate	8
Table 2.2 Explanation of morphemes used on Table 2.1	9
Table 2.3 A list of NE types with the kinds of entities they refer to	10
Table 3.1 Averaged precision results over the medium scaled datasets	13
Table 3.2 Best classification performances of different language model types	15
Table 3.3 Classification accuracy of different classifiers	16
Table 3.4 The effect of the word-based language model on SBD	19
Table 3.5 The effect of the morphological language model on SBD	20
Table 3.6 Rewrite rules in first step and application examples	23
Table 3.7 Some of the word property flags with example words	23
Table 3.8 Example word stems that are correctly detected by Snowball stemmer	26
Table 3.9 Example usage of contextual model for unknown words	30
Table 3.10 Tested feature description	33
Table 3.11 F-measure in MUC metrics on feature related experiments	34
Table 3.12 Possible questions derived from example sentences	38
Table 4.1 Course document classification experiment results (10% training – 90% test)	44
Table 4.2 Course document classification experiment results (50% training – 50% test)	46
Table 4.3 Most distinctive 10 words for classification task based on their IG scores..	49
Table 5.1 Example pre-defined rules about sentence boundary conditions	51
Table 5.2 Regular expressions defined and used in SBD module	52
Table 5.3 Heading and Paragraph class fields with their types	59
Table 5.4 Suggestive numerical values derived from SBD and heading detection experiments	63
Table 5.5 Precision and recall values derived from SBD and heading detection experiments	63
Table 5.6 Experiment results of itemized text parts detection and join operations ...	65

Table 6.1 Main FSM elements and their purposes	68
Table 6.2 Rules to diversify suffixes regarding the first vowel of the suffix	70
Table 6.3 Description of State class fields	70
Table 6.4 Description of Suffix class fields	71
Table 6.5 Description of Transition class fields	72
Table 6.6 Description of SuffixInWord class fields	72
Table 6.7 Description of Path class fields	73
Table 6.8 Phonetic rules considered in the program	78
Table 6.9 Auxiliary terms to control phonetic rules	79
Table 6.10 Methods defined for phonetic rule controls	79
Table 6.11 Exceptional verb roots for simple present tense conversion	80
Table 6.12 Defined suffixes and their formats	80
Table 6.13 Phonetic rule control methods used for each suffix in sequence	83
Table 6.14 Experiment results of verb polarity detection (classification) and conversion model	87
Table 7.1 Categorized tokenizer labels	91
Table 7.2 Search patterns of a 7-token sentence for n-gram lexicon lookups	96
Table 7.3 Lexical (L) and contextual (C) model labels	96
Table 7.4 Defined NE types with their explanations	98
Table 7.5 Categories with most terms for each coarse grained named entity labels ..	101
Table 7.6 Distribution of categories based on the number of terms they contain	102
Table 7.7 Suggestive numerical values derived from NER model experiments	109
Table 7.8 Precision and recall values derived from NER model experiments	109
Table 7.9 NER model experiment results for individual NE types	110
Table 8.1 General approach for true – false question generation using verb polarity information	114
Table 8.2 Specified question type probabilities if value of ContainsTerm property is True	118
Table 8.3 Specified question type probabilities if value of ContainsTerm property is False	118

CHAPTER ONE

INTRODUCTION

1.1 Brief Description and Goals of Thesis

In today's world, information technologies have spread over to all levels of business sectors and operational systems. Using them on educational issues is also a common approach, as they provide considerable advantages such as durability, flexibility, equality of opportunity via easier access and decent gain of time. In line with this purpose, developing a user-friendly and goal-oriented educational software is aimed. What is tried to be achieved by this project within the thesis is analyzing the text-based lecture notes provided by the user to derive reasonable and meaningful questions. Included domains for input lecture notes are history and geography, included question types are true-false, fill in the blanks and multiple choice. Under favour of the generated questions, it is projected to provide an opportunity for students to test and evaluate themselves and make progress on the courses or topics they need. Designed system also allows users to keep track of their progression in time and save the questions they selected. By collecting the meaningful and qualified questions in time, to create a question bank for further usage and allow users to benefit from this service is the long-term goal of the project. To meet these expectations, this research suggests natural language processing (NLP) methods for Turkish language.

1.2 Brief Overview of NLP and Turkish Language

NLP is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages. Therefore, it can be defined as the art of solving (engineering) problems that need to analyze or generate natural language text. To understand the given input text in a language and propose solutions for different NLP task, different analysis levels are required:

- **Phonology:** Concerned with how speech sounds are organized in a given language.

- **Morphology:** Concerned with how words are constructed from primitive units of meaning, which are called *morphemes*.
- **Syntax:** Concerned with the structural relationships between words to form phrases and sentences.
- **Semantics:** Concerned with the denotation of words or phrases within a context and how they combine to form sentence or document meaning.
- **Pragmatics:** Concerned with the different usages and interpretations of sentences.

This research mostly deals with tasks in the scope of former three analysis levels for Turkish language.

Turkish is an agglutinative language (like Finnish, Hungarian and Estonian). In these languages, new words are primarily formed by adding suffixes or prefixes (called affixes in general) to a root word. This might lead to relatively long words, as Turkish can have words with 9 or 10 affixes, while English doesn't tend to stack more than 4 or 5 affixes (Jurafsky & Martin, 2000). In these cases, a Turkish word is frequently equivalent to a whole sentence in English. Figure 1.1 shows how the Turkish word “*yapabileceksek*”, which means “if we will be able to do (something)” in English, is parsed into 5 morphemes (root and 4 suffixes).

TR	yapabileceksek	→	yap	abil	ecek	se	k
ENG	If we will be able to do (something)		do (verb root)	be able to	will	if	we

Figure 1.1 An example Turkish word parsed into its morphemes

As Figure 1.1 shows, suffixes provide additional meanings like subject, tense, aspect (completed, still in progress etc.), polarity (affirmation or negation), mood (is necessary, possible, suggested or desired) when they are appended to a verb root. This is a key characteristic to be considered while selecting proper NLP algorithms to use.

NLP algorithms are divided into three basic models: Statistical, rule-based and hybrid approaches. Main paradigm of statistical models is to automatically learn rules of a language through the analysis of a large corpus of typical real-world examples by dividing them into training and test data. The more user provides examples to train the system, the more reliable output from the test data can be derived. Rule-based models on the other hand, relies on pre-defined grammatical rules about the source language to find out solutions. Hybrid models aim to exploit advantages of both statistical and rule-based approaches with a combined structure. Turkish language suits well with rule-based models as its agglutinative nature leads to specific grammatical rules about suffixes (like which of them may follow which other, which of them may be appended to which root types etc.). In this research, generally rule-based methods are preferred, but statistical methods are also used in some cases.

1.3 Task Distribution of Thesis

The workflow of the test generation process starts when a user loads desired text-based lecture notes in Turkish to the system, then specifies constraints like number of questions and question types to include. Thus, a proper test is generated. So, automatized question and test generation is the primary task of thesis. However, there exist four sub-tasks as pre-requisites to be carried out to fulfill primary task which are document classification, detection of sentence boundaries and headings, detection and conversion of verb polarity and named entity recognition (NER) to detect terms and construct glossary of terms structures for history and geography domains.

1.4 Development Environment of Thesis

Proposed educational software and all sub-modules within the thesis are developed as Windows Forms Application in Microsoft .NET Visual Studio 2017 environment by using .NET framework 4.6.1 and C# programming language. MS SQL Server is used for data storage and management purposes.

Three external libraries are utilized on document classification phase. Open source Turkish NLP framework Zemberek is used for stemming purposes. Data mining and

machine learning library Weka is used to perform experiments. IKVM.NET, which provides .NET implementation of Java class libraries is also included to migrate required Weka libraries to .NET platform, as Weka is developed in Java language.

Another external library named iTextSharp, which provides PDF generation and manipulation functionalities on .NET projects, is used within test generation phase. Using this library, users are allowed to get a single PDF document version of generated tests for a neat and printer-friendly view.

1.5 Organization of Thesis

This thesis is divided into 9 chapters and 5 appendices. Brief description of the thesis, its scope and task distribution are given in Chapter 1. Tasks within the thesis are briefly explained in Chapter 2. Previous academic studies and research on related subjects are mentioned in Chapter 3. Document classification, detection of sentence boundaries and headings, detection and conversion of verb polarity, NER to detect terms and construct glossary of term structures, and automatized question and test generation phases are detailed in chapters 4, 5, 6, 7 and 8 respectively. Finally, a brief summary of the complete thesis is given in consideration of the derived results in Chapter 9.

CHAPTER TWO

TASK DEFINITION

Proposed framework is comprised of four NLP-related sub-modules that make the primary examination module practicable when combined. This makes a total of five tasks in the scope of thesis, which are briefly explained in this section.

2.1 Document Classification

Given a set of classes, document classification (or text classification, text categorization) seeks to determine which class an input document belongs to. Automatic spam detection, sentiment detection, personal email sorting and vertical search engines are some of the real world applications that benefit from classification task (Manning, Raghavan & Schütze, 2009). In statistical text classification, a dataset of documents is divided into training and test sets and documents in training set are labeled to indicate their class information. Decision criterion of the text classifier is learned automatically from training data and experiments are performed on test set. Figure 2.1 represents of an example document classification operation.

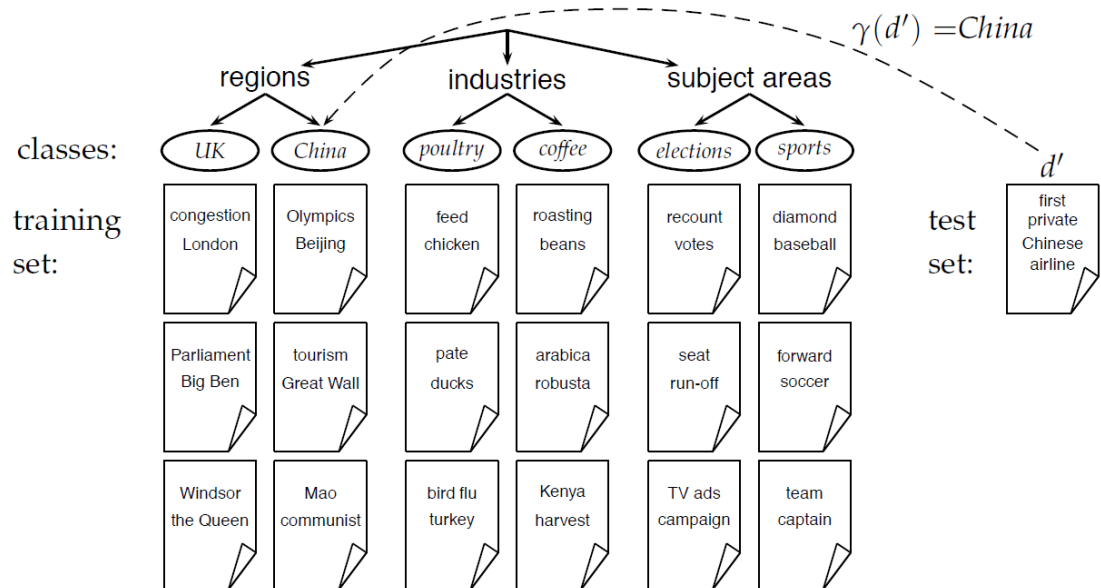


Figure 2.1 Example document classification setup with six classes, each with three training documents and a test set document

Using a weighting scheme, words (or tokens) of documents are converted into feature vectors to represent documents and train the system. Tf-idf, which exploits term frequency (number of times a token occurs in a document) to get term weight information and inverse document frequency (inverse function of the number of documents in which a token occurs) to get term specificity information is a widely preferred scheme, as it reflects the importance of a token to document. Tf-idf formula is given below. (t : term, d : document, D : corpus (dataset), N : total number of documents in the corpus, $tf_{t,d}$: term frequency of term t in document d , df_t : document frequency of term t)

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t} \quad (2.1)$$

Selection of the classification algorithm has direct effect on the success rate, but many other approaches like document standardization (in terms of text file format and used character set), document preprocessing (removal of punctuation marks, eliminating stop words), using n-gram tokens, stemming or lemmatization, using feature selection methods (to reduce number of features) might also have an impact on accuracy, depending on the used dataset and included domains.

In this thesis, document classification is used to automatically detect the domain (history and geography) of an input document, which is a beneficial approach for filtering glossary of terms on test generation phase, filtering existing exams on test selection phase and filtering exam results on test evaluation phase.

2.2 Detection of Sentence Boundaries and Headings

Sentence boundary detection (SBD) is the task of automatically dividing a stream of text or speech into grammatical sentences and a prerequisite for proper sentence processing, as further syntactic and semantic analysis are dependent on this task (Tür, Hakkani-Tür & Oflazer, 2003). Rule-based or machine learning approaches are used on different SBD studies. In formal text, punctuation, capitalization and usage of whitespace characters are utilizable to detect common patterns and generate rules to

state sentence boundary conditions, while speech recognizer output lacks these textual cues and highly depends on machine learning approaches, where actual sentence boundaries are labeled on training data. Identification of abbreviations, initials, ordinal numbers, fractions are important sub-tasks as these expressions mostly contain a period within, which might be considered as a sentence boundary incorrectly. To overcome obstacles arising from abbreviations, using abbreviation lists about source language or automatically detecting abbreviations before SBD (Kiss & Strunk, 2006) are two studied approaches.

Detection of headings can be considered as a task in the scope of classifying document text data into separate sections, but it is also an essential requirement for SBD, especially for studies on raw text data where distinctive tags to separate headings and actual text are not existent. For example, correctly detecting headings with ordinal numbers (like “2. Task Definition”) prevents incorrect sentence boundary detections, when pre-defined rules are not single-handedly sufficient.

In this thesis, sentence boundary detection is the primary task to obtain meaningful sentences and put them into service for other tasks. Heading detection on the other hand aims to detect all primary and sub-headings within a document and assign a heading for each sentence or combine a sub-heading with sentences in necessary cases.

2.3 Detection and Conversion of Verb Polarity

Verb polarity detection (in terms of morphology, not semantics) for Turkish is a sub-task of stemming via morphological analysis, as a negation suffix (*-m/-me/-ma*) is appended to a verb stem before tense, aspect and mood suffixes on negative predicates in most conditions, differently from affirmative predicates. Therefore, successfully parsing a predicate word into its morphemes implies the polarity information.

Stemming and morphological analysis accuracy is crucial to convert a verb to its opposite polarity, as this task remodels the suffix sequence appended to detected stem by transforming required suffixes. Classification of detected stem as verb typed or noun typed is also important, as it is possible to obtain more than a single morpheme

sequence for the same word and this information is essential to eliminate irrelevant morphologic analysis results. Besides, Turkish phonetic rules based on vowel harmony, consonant harmony and root deformation conditions must be considered on this process, as alteration on characters might be required both for stem and suffixes.

While Table 2.1 shows different suffix sequences appended to the same verb root “koş” (run) to form a predicate for both polarity conditions, derived morphemes are explained on Table 2.2. Predicates with different polarity but same tense, person, aspect and mood suffixes are paired up with same index values. Negation suffixes on morpheme sequences of predicates with negative polarity are emphasized with bold. As it is seen, if negation suffixes are opted out, morpheme sequences look alike the sequence of corresponding affirmative predicate with same index value. Changes on some characters arise from Turkish phonetic rules, which will be explained in detail on Chapter 6.

In this thesis, a finite-state machine (FSM) structure is constructed for verb stemming purpose and detection of verb polarity is an embedded task in this module. After all sentences obtained from SBD module are processed through FSM, verb polarity conversion operations are applied on a sentence only if morphological disambiguation is achieved. Main goal of this complete process is to generate a true – false question for the processed sentence.

Table 2.1 Changes on morpheme sequences based on the polarity condition of a predicate

Predicate Index	Turkish Predicate	English Meaning	Polarity	Morpheme Sequence
1	Koştum	I ran.	Affirmative	koş – tu – m
1	Koşmadım	I didn’t run.	Negative	koş – ma – dı – m
2	Koşuyor	He/She is running.	Affirmative	koş – uyor
2	Koşmuyor	He/She is not running.	Negative	koş – m – uyor
3	Koşmalıyız	We should run.	Affirmative	koş – malı – yız
3	Koşmamalıyız	We should not run.	Negative	koş – ma – malı – yız
4	Koşalım	Let’s run.	Affirmative	koş – a – lım
4	Koşmayalım	Let’s not run.	Negative	koş – ma – ya – lım

Table 2.2 Explanation of morphemes used on Table 2.1

Predicate Index	Polarity	Stem Morpheme	Suffix Morphemes
1	Affirmative	koş	<i>tu</i> (past tense with -di) – <i>m</i> (1 st person singular)
1	Negative	koş	<i>ma</i> (negation) – <i>dı</i> (past tense with -di) - <i>m</i> (1 st person singular)
2	Affirmative	koş	<i>uyor</i> (present tense)
2	Negative	koş	<i>m</i> (negation) – <i>uyor</i> (present tense)
3	Affirmative	koş	<i>malı</i> (necessitative) – <i>yız</i> (1 st person plural)
3	Negative	koş	<i>ma</i> (negation) - <i>malı</i> (necessitative) – <i>yız</i> (1 st person plural)
4	Affirmative	koş	<i>a</i> (optative) – <i>lim</i> (1 st person plural)
4	Negative	koş	<i>ma</i> (negation) – <i>ya</i> (optative) – <i>lim</i> (1 st person plural)

2.4 Named Entity Recognition to Detect Terms

The term named entity (NE) is used to define anything that can be referred to with a proper name. The process named entity recognition (NER), which is a subtask of information extraction, aims to locate and classify named entities in text into pre-defined categories. This is a combined task, as it must fulfill two requirements respectively: To find bounds of text that constitute proper names and to classify them according to their types correctly.

Generic news-oriented NER systems focus on detecting expressions that indicate people, places and organizations, while specialized applications may be concerned with many other types of entities, including commercial products, works of art, proteins, genes and other biological entities (Jurafsky & Martin, 2009). In most NER systems, it is a common approach to extend the scope of a NE to include things that aren't proper names but have characteristic meanings within the text. This generally leads the inclusion of temporal expressions like dates, times, named events and numerical expressions like dates, times, named events and numerical expressions like

measurements, counts, prices to the NE categories (also called as tags). Table 2.3 shows example NE types and possible instances in the scope of these types.

Table 2.3 A list of NE types with the kinds of entities they refer to

NE Type	Tag	Sample Categories
Person	PER	Individuals, fictional categories, small groups
Organization	ORG	Companies, agencies, political parties, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-Political Entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicle	VEH	Planes, trains, automobiles

NER systems mostly take an unannotated block of text as input and produce an annotated block of text that points the names of named entities. For example, the projected output for the unannotated input text “*Mustafa Kemal Atatürk 1881 yılında Selanik’te doğdu.*” (Mustafa Kemal Atatürk was born in Thessaloniki in 1881.) is “[*Mustafa Kemal Atatürk*]_{Person} [*1881*]_{Date} *yılı*_{nda} [*Selanik*]_{Location} *’te doğdu.*”

Word ambiguity is a major concern for NER systems, like most of the other natural language processing (NLP) tasks. For example, the word “Washington” might indicate a person, a location, an organization (a sports club) or a facility (a ship). Or the word occurrence “Ural” in Turkish text can refer to a location (a river) or a person.

NER algorithms are divided into three models: Statistical, rule-based, hybrid approaches. Main paradigm of statistical models is to automatically learn rules and patterns of named entities through a pre-annotated training data. Besides, training data has to be labeled to provide information about selected features if used. Most common statistical models are Hidden Markov Models (HMM), Maximum Entropy (ME) and Conditional Random Fields (CRF). Rule-based models rely on orthographical, morphological and lexical information derived from feature sets. Syllabication, tokenization, morphological analysis or lexicon lookups are the main operations to assign feature values. Using lexicons to store person, location and organization names that imply a NE existence is a common approach. Pre-defined grammatical rules and character transformation conditions about the source language are also beneficial,

especially for agglutinative languages which require intensive suffix usage. Hybrid models aim to exploit advantages of both statistical and rule-based approaches with a combined structure. It is a serviceable approach to reduce effects of domain changes, but storage requirements and possible system overhead should not be neglected.

A NER system with high success rate might be serviceable for many applications and use case scenarios in today's world, like classifying content for news providers, recommender systems, customer support, media analysis, sentiment detection, email scanning, more accurate literature search or educational purposes which the proposed NER model within this thesis is developed for. This model is specialized for Turkish lecture notes within history and geography domains to detect named entities. Detected characteristic terms are used as sources to build glossary of terms structures for geography and history domains, which are used on question generation phase.

2.5. Automatized Question and Test Generation

Four sub-modules with different NLP tasks developed within this thesis are combined to carry out automatized question and test generation on input text-based lecture notes provided by the users, which is the main educational task. Users are also allowed to specify constraints about the test to be generated, which are number of questions, included question types, preserved or shuffled sentence order. System is specialized for history and geography domains. True - false, fill in the blanks and multiple choice are the question types enabled for selection. Figure 2.2 shows the relationships between the tasks defined within this thesis.

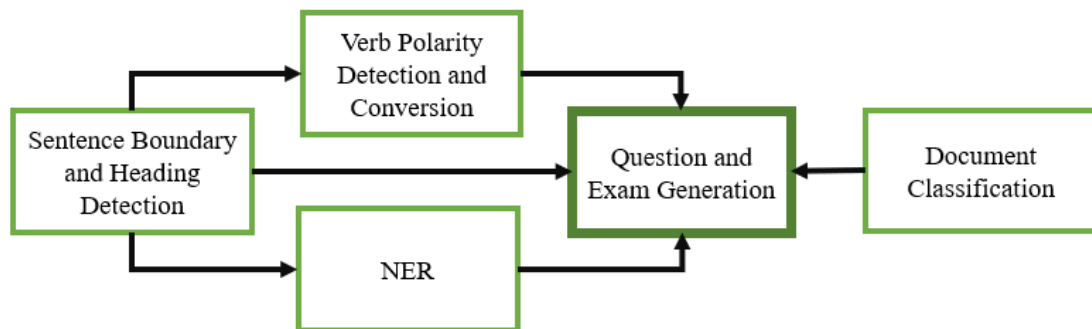


Figure 2.2 A diagram showing relationships between the tasks defined within the thesis

CHAPTER THREE

PREVIOUS WORK

As stated previously, developed examination system is a combination of different modules, each with a different NLP task to deal with. Therefore, this study touches upon document classification, sentence boundary detection (SBD), stemming (with morphological analysis), named entity recognition (NER) subjects alongside exam and question generation. A comprehensive literature review is made to cover all.

3.1 Document Classification

Lewis (1992) investigated the effect of feature set size for word-based indexing for text categorization task on Reuters and MUC-3 data sets and optimal feature set size is identified as 10 to 15. Used statistical model is evaluated by breakeven point metric, the highest value that precision and recall are equal and 0.65 is reached on Reuters data set with 10 features. Besides, word features are found out to be more effective than extracted features by syntactic analysis and feature clustering.

Fürnkranz (1998) investigated the effect of using n-gram words features for text categorization, after removal of stop words. Study revealed that word sequences of length 2 or 3 is the most beneficial, while using longer sequences reduces classification performance. Besides, it is stated that unigrams give higher precision than multi-grams. Inductive rule learning algorithm Ripper is used for experiments in news domain and 81.18% is the highest f-measure value recorded.

Han & Karypis (2000) determined that using a centroid-based classifier with averaged similarity measure for each pre-defined document category (using weighted tf-idf representation to represent a document) is a preferable method for document classification. In 17 out of 23 experiments with different data sets, centroid-based classifier was superior to Naïve Bayes (NB), k-nearest neighbor (kNN) and decision tree algorithms.

Manevitz & Yousef (2001) implemented a one-class support vector machine (SVM) for document classification task and compared it with other one-class classification algorithms. Experiments on Reuters data set showed that SVM approach outperformed all methods except neural network classifier. SVM performance also turned out to be sensitive to changes on selected data representation and kernel function.

Slonim, Friedman & Tishby (2002) proposed a greedy sequential information bottleneck (sIB) clustering algorithm to unsupervised document classification. Experiments are performed on small and medium scaled datasets collected from 20Newsgroups and Reuters-2000 corpora. Algorithm resulted in 83.3% precision on small-scale dataset and 76.6% precision on medium-scaled dataset on average. It is stated that sIB outperformed other clustering algorithms, even seemed to be almost competitive with supervised NB classifier. Table 3.1 shows the results over medium scaled datasets.

Table 3.1 Averaged precision results over the medium scaled datasets (Slonim et al., 2002)

Dataset	sIB	sK-means	K-means	sL1	sKL	NB
NG10	79.5	76.3	70.3	27.7	58.8	80.8
NG20	57.5	54.1	53.4	15.3	28.8	65.0
Reuters	85.8	64.9	66.4	70.1	59.4	90.8
New-Reuters	83.5	66.9	67.3	73.0	81.0	85.8
Average	76.6	65.6	64.4	46.5	57.0	80.6

Amasyalı & Diri (2006) worked on text classification for Turkish using n-gram model. To evaluate system success, three tasks are defined as detecting the author, genre and the gender of author of a text document. Used dataset contains 630 newspaper articles written by 18 authors on 3 different subjects (political, popular interest and sport). 83.3% by NB for author identification, 93.6% by SVM for genre identification and 96.3% by SVM for gender identification are the highest success rates.

Yıldız, Gençtav, Usta, Diri & Amasyalı (2007) proposed to exploit Turkish morphology and use word stems instead of a bag-of-words approach for text

classification task. Used dataset contains newspaper articles from three daily Turkish newspaper and categorized in five classes (economy, health, magazine, sport and politics). Turkish NLP library Zemberek is used for the stemming phase. Words that occur only once on the entire dataset are excluded when selecting feature vectors. Experiments showed that using stemming mostly improves the success rate. Highest accuracy score is stated as 96.25% via NB classification algorithm.

Kesgin (2007) developed correlated stemmer and text classifier structures for Turkish text documents. It is stated that Turkish phonics rules might require character level transformation for a more accurate stemming operation. Developed software allows users to load training documents and define categories to be used for classification. Vector representations for words that are derived after training phase are stored in database for further operations.

Isa, Lee, Kallimani & RajKumar (2008) proposed a hybrid text classification method that uses Bayes formula to derive vectors that represents a document, then uses SVM to classify the documents. This method reported significant reduction in training time and improvement in classification accuracy compared to single NB or TF-IDF/SVM hybrids on 20Newsgroups, Vehicles (Wikipedia), Automobiles (Wikipedia) datasets. However, NB outperformed NB-SVM hybrid on Mathematics (Arxiv.org) dataset, on which defined classes share many common keywords.

Amasyalı & Beken (2009) proposed to locate words on a multi-dimensional semantic space and use this vector space for text classification task. Their study based upon the hypothesis that two words' semantic similarity is related with the number of documents which the words co-occur. 15K web pages are scanned to locate approximately 4500 different stems on space. Best results are reached when linear regression is used for classification on 100-dimensional space with 93.25% accuracy. Figure 3.1 shows distribution of news text in 5 categories (economy, magazine, health, politic, sport) on first two dimensions.

Tantuğ (2010) focused on document categorization for agglutinative languages with statistical models and compared different approaches like using standard word forms,

root forms, root forms with part-of-speech (POS) info, truncated word forms and character sequences. Dataset contains 20K news documents in eight categories. Using truncated word forms (when first 4 characters of each word are taken) resulted in 81% and character based modeling resulted in 82% f-measure on their best individual performances and proven to be preferable methods to deal with possible data sparseness problems on agglutinative languages. Best f-measure values obtained from tested approaches are shown on Table 3.2 (n denotes language model order parameter).

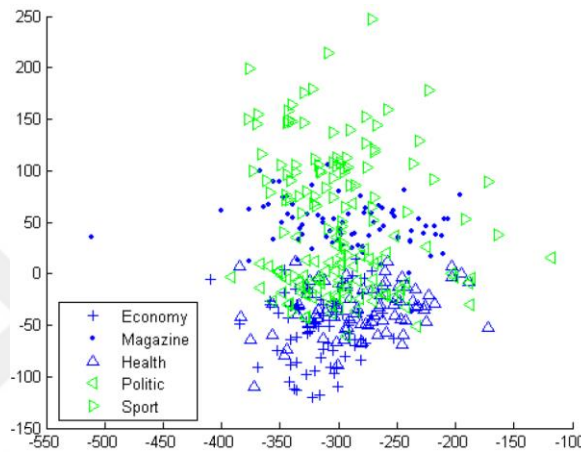


Figure 3.1 Distribution of news text on first two dimensions

Table 3.2 Best classification performances of different language model types

n	Word form	Root	Root + POS	First4Char	CharBased
1	0.7721	0.8017	0.7749	0.7924	0.6023
2	0.7767	0.8181	0.7891	0.8133	0.7488
3	0.7772	0.8192	0.7838	0.8139	0.8078
4	0.7772	0.8192	0.7859	0.8134	0.8212
5	0.7769	0.8192	0.7850	0.8134	0.8209

Ting, Ip & Tsang (2011) inspected the sufficiency of NB as a document classification algorithm. Experimental dataset contains 4000 documents equally divided into four categories (business, politics, sport, travel) and %30 of them is used for training. NB turned out to have best accuracy result with SVM, compared to neural network (NN) and decision tree (DT). NB classifier is better in terms of computational efficiency though, as it required only 0.19 seconds to build the model, while SVM required 2.69 seconds. Table 3.3 shows the classification results of different classifiers.

Table 3.3 Classification accuracy of different classifiers

	Correctly Classified Instances	Incorrectly Classified Instances	Precision	Recall	F-Measure
Naïve Bayes	2717 (97.0%)	83 (3.0%)	0.970	0.970	0.970
SVM	2712 (96.9%)	88 (3.1%)	0.969	0.969	0.969
NN	2605 (93.0%)	195 (7.0%)	0.931	0.930	0.930
DT	2551 (91.1%)	249 (8.9%)	0.911	0.911	0.911

Uysal, Günal, Ergin & Günal (2012) developed an SMS spam message filter application using text and pattern classification techniques and effective feature selection. kNN is used as classifier, while Gini index is the chosen approach for feature selection. For training, a database that contains English SMS messages (747 spam and 4827 normal) is used. Experiments showed that using only the 10 most distinctive terms as feature set gave the best results.

Tüfekçi, Uzun & Sevinç (2012) proposed to exploit Turkish grammatical rules to reduce the dimension of feature vector without decreasing text classification success. Experiment results on web based news articles indicate that including only noun typed word stems to feature set reduced the initial dimension size by %97.46 and reached to 92.73% accuracy with NB usage.

Uysal & Günal (2014) proposed latent semantic features (GALSF) to represent documents in text classification and latent semantic indexing (LSI) for feature transformation approach. Expected contribution from LSI is stated as revealing underlying hidden concepts such as synonym and polysemy while transforming term-document matrix into a new subspace.

Çoban, Özyer & Özyer (2015) worked on a dataset of Turkish Twitter messages and applied document classification methods to correctly guess whether the sentiment of context is positive or negative. Differently from other studies, extraction of recurrent characters in a word for normalization is applied on preprocessing phase. For example, the word “*günaayduunnn*” is translated to “*günaydın*”. System also benefits from

emoticon usage. Best accuracy score is recorded as 66.06%, which is achieved by the n-gram model with multi-nominal NB (M-NB) as classifier.

Yang et al. (2016) proposed a hierarchical attention network (HAN) to reveal more informative words to be used as features for document classification. Their study based on the observation that same word or sentence may be differentially informative in different documents, so system uses context unlike a bag-of-words approach. Datasets consist on user reviews (On Yelp, IMDB, Yahoo Answers, Amazon) and system is expected to guess the given rating score (5 or 10 classes on each dataset). Best accuracy is reached on Yahoo Answers dataset (with 10 classes) with 75.8%.

Yıldırım & Yıldız (2018) compared traditional bag-of-words approaches to neural network language models (NNLM) for Turkish text classification task. Considering experiment results, it is stated that a bag-of-words model utilized with an effective feature selection reaches comparable performance with NNLM. Using Information Gain (IG) or chi-square (X2) as feature selection method with M-NB algorithm is indicated as a successful combination.

3.2 Sentence Boundary Detection

Riley (1989) described a regression tree model for many pattern recognition and natural language processing problems. End of sentence detection for English is one of the tasks. The problem is defined as classifying a period as "end of sentence", "end of abbreviation" or both conditions at the same time. Model is constructed with 25 million words of news text sources and resulted in 99.8% accuracy on Brown corpus.

Aberdeen et al. (1995) introduced Alembic system, which benefits from rule sequences for many tasks. SBD is defined as a supportive zoner task for part-of-speech (POS) tagging task in UNIX preprocess module. *Punctoker* (to find word boundaries that are not whitespace), title-tagger (to mark personal titles and honorifics) and *parasenter* (to zone text for paragraph and sentence boundaries) are the main preprocessors used for SBD on the system.

Palmer & Hearst (1997) presented an adaptive multilingual sentence boundary model. Offered system, called Satz, uses a small lexicon and training corpus to estimate the POS distribution of surrounding words of a punctuation mark. Neural network and decision tree are the used algorithms to classify the punctuation mark. System is tested on Wall Street Journal (WSJ) corpus for English and best results gave near 99% accuracy.

Reynar & Ratnaparkhi (1997) presented two Maximum Entropy (ME) models to detect sentence boundaries. Both models seek to classify each occurrence of candidate characters {., ?, !} as valid or invalid sentence boundary. First model is targeted at high performance for English and uses some language specific features, while second model is aimed at being more portable across languages. System automatically produces the abbreviation list from training data and uses this list to provide contextual information for contextual features. Portable model tests reached 98% accuracy on WSJ corpus and 97.5% accuracy on Brown corpus.

Temizsoy & Çiçekli (1998) developed an ontology-based approach for Turkish sentence parsing task. System also benefits from morphological marks (suffixation) of Turkish to handle ambiguity problems on text. Using these rule-based resources, methodology aims to detect interaction between syntactic and semantic information instead of building a syntactic tree structure. Used ontology represents entities (atemporal individuals), events (temporal phenomena) and relationships between them in a hierarchical structure.

Stamatatos, Fakotakis & Kokkinakis (1999) used transformation-based learning (TBL) to extract sentence boundary rules automatically. System is trained with a corpus of Greek newspaper articles. In the first learning stage, it is assumed that all candidate sentence-ending punctuation mark is actually a sentence boundary. Then, using characteristics like capitalization of the token with possible end-of-sentence boundary marker and the following token, triggering conditions to remove a sentence boundary are tried to be detected. In the second stage, triggering conditions to insert a sentence boundary are searched. System produced an error rate of 0.6%.

Gotoh & Renals (2000) are worked on SBD in broadcast speech transcripts. They introduced an n-gram language model via sentence boundary information derived from finite state models, and an alternative model estimated from pause duration information derived from speech recognizer outputs. Later model outperformed the former one while combination of two improved precision and recall scores over 70%.

Mikheev (2000) proposed a model that combines SBD, proper name identification and abbreviation detection in one system. Additionally, the model treats the SBD problem as a sub-task of POS tagger which is built as a Hidden Markov Model (HMM) and ME combination. Main idea on this study is to classify a candidate upper-cased token as a proper name or an abbreviation based on instances of that type within an unambiguous context; which would also help to disambiguate sentence boundaries.

Tür et al. (2003) used a statistical HMM with two inner models to handle SBD task. System doesn't benefit from punctuation marks considering the usability of speech recognition output. Task is considered as a boundary classification problem, so each word boundary in training set is labeled as sentence-boundary (YB) or non-sentence boundary (NB). Word-based model uses surface forms of words and checks the probability of sentence boundary between words, while morphological model benefits from final inflectional groups derived from morphological analyses of words and checks the probability of sentence boundary after an inflectional group sequence. When system is trained with 18 million words from Milliyet newspaper dataset and both models are included, error rate is calculated as 4.34%.

Table 3.4 shows probability values of being NB and YB between words “*geldi*” (came) and “*çünkü*” (because) according to the word-based model. Based on these values, it can be said that it is 30 times more possible to have a sentence boundary between these 2 words.

Table 3.4 The effect of the word-based language model on SBD

Output sequence	Probability
geldi NB çünkü	0.00028166
geldi YB çünkü	0.00614714

Table 3.5 shows probabilities of being NB and YB after the word “geldi” (came) according to the morpheme sequence obtained by morphological model (*Pos* indicates positive polarity, *A3sg* indicates third person singular agreement).

Table 3.5 The effect of the morphological language model on SBD

Output sequence	Probability
Verb+Pos+Past+A3sg NB	0.24849
Verb+Pos+Past+A3sg YB	0.751505

Wang & Huang (2003) introduced Bondec system, which consists of three independent applications (rule-based, HMM and ME) for SBD task. Annotated train and test data, which are constructed from WSJ corpus are obtained from Palmer & Hearst's (1997) study. System also uses lexical resources for common last names, first names and honorifics; but automatically extracts abbreviations. Rule-based model gave best results for precision with 99.56%, but uncommon cases led a lower recall resulted in 76.95%. ME model is defined as the central part of the system and reached best results among the three with an error rate less than 2%, using eight binary features; while HMM resulted in 10%.

Dinçer & Karaoğlu (2004) used Turkish syllabication and phonetic rules collectively to disambiguate dots that indicates an end-of-sentence (EOS) from the ones that are used for other purposes. Rules are represented as trigram combinations which includes a dot and its adjacent character sequences. For example, [W * W] which is one of the ambiguous cases, denotes the situation where a letter sequence W which starts with an uppercase character, is followed by a dot and then followed by a letter sequence W which starts with an uppercase character. Syllabication is proposed to detect abbreviations and make progress on disambiguation. As a Turkish word may be composed of a sequence of one or more six predefined syllable patterns: V, VC, VCC, CV, CVC and CVCC (C indicates a consonant, V indicates a vowel); it is stated that if a dot follows an abbreviation, the sequence is expected not to be a valid Turkish word hence it does not have a valid syllabication. Test results show that proposed system reached 96.02% accuracy.

Kiss & Strunk (2006) presented Punkt system, a language-independent model for SBD, which is also one of the most successful approaches for Turkish. Main assumption is that most ambiguities in SBD can be overcome once abbreviations have been identified. In this direction, system mainly tries to detect abbreviations by using log-likelihood ratio algorithm. System also shows the potential of detecting initials and ordinal numbers as subtasks. Newspaper corpora for eleven languages are used for system evaluation and METU Turkish Corpus is the one used for Turkish. For Turkish, system accuracy reached up to 98.69%, while mean accuracy for eleven languages is calculated as 98.74%.

Liu & Shriberg (2007) compared alternative evaluation metrics (like classification error rate per word boundary, precision-recall (PR) curves, ROC curves, area under the curves etc.) for SBD task instead of using a single error metric. The study showed that decision curves might provide useful information to choose more preferable models for specific regions. Another finding is that using PR curves for an imbalanced data set generally provides better visualization for viewing differences among different algorithms.

Güz, Favre, Hakkani-Tür & Tür (2009) introduced generative, discriminative and hybrid classification methods and lexical, morphological and prosodic features for Turkish SBD task on speech data. Turkish broadcast news speech corpus collected at Boğaziçi University BUSIM Laboratory is used for experiments. System used about 200 word-level prosodic features like pause duration at boundary and normalized phone durations of the word preceding the boundary. When conditional random fields (CRF) using all features is combined with factored hidden language modeling (fHELM), system reached 0.926 f-measure value.

Read, Drizan, Oepen & Solberg (2012) evaluated several publicly available SBD systems both in edited, formal language text and user-generated web content in English. As expected, decrease on success rates of all tested systems is observed. Using unsupervised learning combined with heuristic rules, also tools to automatically acquire domain-adapted lists of abbreviations is proposed for the future work.

Aktaş & Çebi (2013) described SBD task as the process of generation a corpus. They offered a rule-based sentence detection method for Turkish. System uses 21 sentence boundary rules which are determined by linguists and an abbreviation file which has been taken from Turkish Linguistic Association. Each rule consists of a character that indicates the first character of the word before punctuation mark that is used to end sentences, a character to state the punctuation mark itself and a character that indicates the first character after punctuation mark. System provides success rate in a range of 99.6% and 99.8% on randomly selected columns from two Turkish newspapers.

Xu et al. (2014) worked on SBD in broadcast news. System uses prosodic feature inputs on a deep neural network (DNN) model and maps them into boundary or non-boundary posterior probability outputs. CRF is used to combine these probability outputs with lexical features derived from text and to label inter-word positions as boundary or non-boundary. This DNN-CRF model reached 81% f-measure on reference transcripts (REF) and 64.9% f-measure on speech recognition output (ASR).

Bektaş & Özel (2018) studied on the effects of using POS tag information on SBD task for Turkish. Proposed system uses two features that indicates the POS tag of words before and after the candidate end-of-sentence character along with initial nine features taken from rule-based models. Used dataset is derived from a subset of Turkish National Corpus. Five different supervised learning methods are tested, and experiments showed that including POS tag features increased the success rate on four of them (except Radial Basis Function (RBF) network) and using decision trees gave the best accuracy result with 86.2% (improved from 84.7%).

3.3 Stemming

Porter (1980) introduced one of the most widely used stemming algorithms for English. This study depends on a lexicon-free model and uses a series of rewrite rules for automatic removal of suffixes from words. Algorithm executes on a sequential basis and different sets of rules are controlled on each step. Longest matching rule is

considered if more than one rule is matched. For example, first step contains four rules to normalize plural nouns and third person singular verbs, shown on Table 3.6.

Table 3.6 Rewrite rules in first step and application examples

Rule	Example Application
SSES → SS	caresses → caress
IES → I	carries → carri
SS → SS	caress → caress
S → ε	cares → care

Solak & Oflazer (1993) proposed a morphological root-driven parser for a Turkish spelling checker module. System is provided with a dictionary of about 23.000 words, which contains root morphemes and some irregular stems based on Turkish New Writing Guide. 41 flags, that indicates certain word properties are used to detail each dictionary entry. Some of the used flags are shown on Table 3.7.

Table 3.7 Some of the word property flags with example words

Flag	Property of the word if flag is set	Examples
CL_ISIM	Is a nominal root	BEYAZ, OKUL
CL_FIIL	Is a verbal root	SEV, GEZ
IS_OA	Is a proper noun	AYŞE, TÜRK
IS_OC	Is a proper noun which has a homonym that is not a proper noun	MISIR, SEVGİ
IS_SD	Is a nominal root ending with a consonant which is softened when a suffix beginning with a vowel is attached	AMAÇ, PARMAK, PSİKOLOG
F_UD	Is a verb root which has a vowel {I} in its last syllable that drops when the passiveness suffix –{I}L is affixed	AYIR, SAVUR

The root of a word is searched in the dictionary using a maximal match algorithm, by removing a letter from the right until a matched substring is found. This approach is backed up with parser execution considering checked flags, as longest matched substring might lead to incorrect roots if single-handedly used. For example, correct root of the word “*yapıldın*” (you were made) is the verbal root “*yap*” (do, make), not nominal root “*yapı*” (structure).

Solak & Can (1994) developed a similar approach for stemming in Turkish, using a lexicon with actively used stems, describing records using 64 tags and searching for

the word stem by pruning a letter from right end at each step, applying morphological analysis for each candidate stem and finally returning a possible stem set for the word.

Xu & Croft (1998) proposed using corpus-based word co-occurrence statistics for stemming and tested the approach on English and Spanish text. Main ideas behind their study is that words with multiple meanings may state a different primary meaning on different corpora and word variants that should be conflated will occur in the same text windows. Used technique aims to prevent irrelevant conflations like “policy / police” and “addition / additive”, as such unrelated words co-occur rarely. Experiment results show that co-occurrence analysis is a good technique to improve average precision of a stemmer.

Cebiroğlu (2002) introduced a rule-based model to find out a Turkish word’s root without using a lexicon. It is stated that Turkish suffix sequences can be defined with stable rules and using these rules, a word can be morphologically parsed to reach its root. Suffixes are divided into five sets (derivational suffixes, name inflectional suffixes, affix-verbs, verb tense suffixes, verb inflectional suffixes) and a finite state machine (FSM) that contains suffix order rules (from end to the beginning of a word) is defined for each set. Separate FSM structures are combined with predefined work order rules. After system execution, possible roots of a word with their type (as noun or verb) are detected.

Oflazer (2003) proposed a dependency parsing model with extended FSM for Turkish by dividing words into inflectional groups (IG) which are separated by derivational boundaries (^DB). A sentence is represented as a sequence of IGs, which are used to define syntactic relation links. It is observed that a link starts only from the last IG of a (dependent) word and land on one of the IGs of a (head) word on the right. 10 syntactic relations are defined on the model: Subject, Object, Modifier (adverb/ adjective), Possessor, Classifier, Determiner, Dative adjunct, Ablative adjunct, Locative adjunct, Instrumental adjunct. Syntactic relation (dependency) rules are stored as regular expressions that indicate the dependent IG, head IG and IGs in between to be skipped over. Figure 3.2 shows dependency links on an example

sentence which is represented as an IG sequence. POS information of each IG is also included (Abbreviations: *Det* for determiner, *Subj* for subject, *Obj* for object, *Mod* for modifier, *Adj* for adjective, *Adv* for adverb, *Pron* for pronoun).

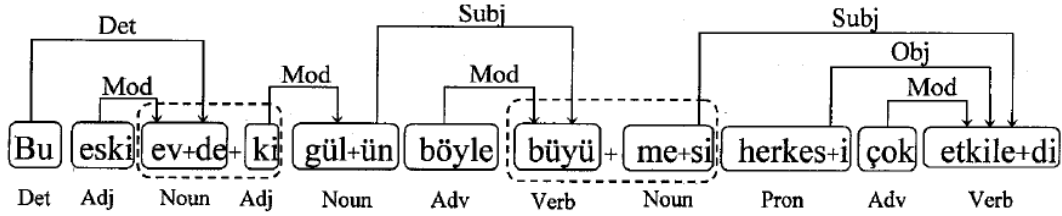


Figure 3.2 Dependency links and POS tags on an example sentence

Sever & Bitirim (2003) introduced FindStem algorithm to find the root, apply morphological analysis and choose the stem of the examined word respectively on three different components. This approach starts the root search phase from the beginning of a word and appends a letter on each step. A lexicon with root words and possible root changes is used as an auxiliary structure. Afterwards, morphological analysis is applied to eliminate irrelevant roots and derive candidate stems. Finally, a word stem is selected among candidate derivations list. To test the algorithm, average number of relevant documents retrieved is compared over stemmed and unstemmed data, when 15 queries are executed on a dataset with about 2500 documents. Unstemmed index resulted in 23.3, while stemmed index outperformed it with 28.4 (350 to 426 respectively in totals).

Dinçer & Karaoğlu (2003) developed a lexicon free, probabilistic stemmer for Turkish, while it is stated as applicable for other agglutinative languages like Finnish, Hungarian etc. Proposed model benefits from probabilities of an ordered pair of letters (h1,h2) being in the stem part, suffix part or between the stem and suffix part of the given word to correctly determine a substring as stem. Experiments are done on Turkish news texts and stemmer achieved to get correct stems with 95.8% success rate.

Çilden (2006) implemented a stemmer for Turkish language using Snowball language which is commonly used to develop stemmers for (mostly agglutinative) many languages. Main focus is to make progress on information retrieval (IR)

purposes, so proposed model is focused on finding noun stems in text and covers only suffixes that are appended to noun stems (which are noun suffixes and nominal verb suffixes). Right-to-left FSM rules to derive noun stems are translated into Snowball expressions to generate stemmer. Table 3.8 represents some successful results.

Table 3.8 Example word stems that are correctly detected by Snowball stemmer

Word	Morphological Analysis	Meaning	Stem
Kalelerimizdekilerden	Kale-lAr-UmUz-DA-ki-lAr-DAn	From the ones at one of our castles	Kale
Çocuğuyumşumcasına	Çocuk-(s)U-(y)mUş-(y)Um-cAsInA	As if I were her child	Çocuk
Kedileriyle	Kedi-lAr-(s)U-(y)lA	With their cats	Kedi
Çocuklarımmış	Çocuk-lAr-(U)m-(y)mUş	Someone told me that they were my children	Çocuk
Kitabımızdı	Kitap-UmUz-(y)DU	It was our book	Kitap

Akın & Akın (2007) introduced open source NLP framework Zemberek for Turkic languages, which provides operations including morphological parsing and stemming. For each language implementation, system uses predefined grammar requirements and language data like alphabet, suffixes, root tree, special root and suffix cases etc. Morphological parser and stemmer operations use a dictionary based top-down approach where root candidates are found firstly, then possible suffix combinations are investigated.

Sak, Güngör & Saraçlar (2008) developed a finite-state implementation of a morphologic parser and applied Viterbi decoding using averaged perceptron algorithm for morphological disambiguator. A lexicon with 54.267 root words is also included using TDK dictionary. Used feature set takes the morphosyntactic information of current and previous two tags to allow left and right Viterbi decoding to find out best morphological parse sequence for a sentence. Accuracy of the proposed disambiguator is calculated as 97.81%.

Aktaş (2010) proposed rule-based methods for different NLP tasks like sentence boundary detection, POS-tagging and morphological analysis. System uses rules to specify which suffixes can be appended to which root - stem types (noun or verb), besides 85 syntactic (word order) rules based on word types. Rules list is stored in

XML format to provide easier modification. Project includes a complete lexicon with the list of words in Turkish dictionary, provided by Turkish Linguistic Association (Türk Dil Kurumu, TDK). Using this lexicon, a database structure is built, and possible words, stems and suffixes are stored in 7 different tables which are: **Kokler** (Roots), **KoklerSanal** (Modified roots by vowel changing rules), **Govde** (Stems), **Kelimeler** (Words), **Grup** (Information about suffix groups), **Ek** (Suffixes), **Ekler** (All suffix versions according to morphophonemic rules).

- **Kokler** table holds Boolean attributes to store information of whether a root can be a noun or not, also verb or not.
- **Govde** table stores a list of stems in Turkish with Boolean attributes to indicate possible POS tag info with related attributes like isNoun, isAdj, isAdv.
- **Grup** table stores the meaning of tags, which are used to define the rules in the process of parsing stem or root and indicates the word types that the suffix group can be added to.

Öztürkmenoğlu & Alpkoçak (2012) compared no-lemmatization approach with three different lemmatization approaches for Turkish (morphological analyzer based on FSM, dictionary-based lemmatization (DTL) and stemmer module of open-source Zemberek tool) and fixed length truncation for information retrieval (IR) over Milliyet newspaper articles collection. Experiment results demonstrated that using lemmatization increases IR performance and using maximum length lemmas instead of minimum is more beneficial. Also, DTL has reached more effective IR performance than other approaches with highest mean average precision (MAP) value in 34.86%.

Şahin, Sulubacak & Eryiğit (2013) introduced a two-level Turkish morphological analyzer based on a lexicon and analyzed the effects of using flag diacritics to deal with exceptions in phonetic and morphological rules. Flag diacritics are beneficial to allow or disallow exceptional conditions or certain affix concatenations which may be impossible or impractical to attain by updating finite state transducer (FST) implementation. Flag diacritic types are defined as unification, positive setting, negative setting, require test, disallow test and clear feature. For example, @U.Hş.var@ and @U.Hş.yok@ are unification type flags that indicate whether a verb

(like “*döv*”, “*göl*”, “*it*”) is allowed to take a reciprocal suffix (-*Uş*) or not (like “*yolla*”, “*salla*”, “*havalan*”).

Moral, Antonio, Imbert & Ramirez (2014) assessed benefits and drawbacks of stemming approaches on IR process on different languages. It is stated that there is not a wide agreement about the usefulness of stemming on IR, as many other factors can have an influence on performance. Also using performance metrics like compress ratio of input vocabulary, types of generated error (like over-stemming and under-stemming) are suggested for a more accurate evaluation instead of using precision and recall metrics. Even so, many researchers agree on benefits of using a stemmer for IR purposes on highly inflective languages (like Turkish), on datasets with short documents or when there are data storage limitations.

Proposed model in this report uses stemming to detect polarity of verbs and accurately convert affirmative to negative or negative to affirmative. Most of the morphological analyzers developed for Turkish are able to detect the polarity information (affirmation or negation) of a verb. However, there is no published study on conversion of polarity between each other for Turkish verbs, as far as we know. Instead, polarity information is widely used for sentiment analysis purposes in the literature.

Vural, Cambazoglu, Senkul & Tokgoz (2012) proposed a framework for unsupervised sentiment analysis in Turkish text documents, using a movie reviews dataset obtained from Beyazperde, a well-known website about movies. Özsert & Özgür (2013) constructed a word relatedness graph by using relations in WordNet and proposed a random walk model using commute time as proximity measure for multilingual word polarity detection. Dehkharghani, Saygin, Yanikoglu & Oflazer (2016) introduced SentiTurkNet, a comprehensive polarity resource for Turkish by assigning three polarity scores for each synset in Turkish WordNet, to designate its positivity, negativity and neutrality levels.

3.4 Named Entity Recognition

Message Understanding Conferences (MUC) are designed to promote and evaluate research in information extraction. These conferences were initiated by NOSC (Navy Operational Support Center) to assess research on the automated analysis of military messages containing textual information. Two primary evaluation metrics precision and recall are detailed and used for IE tasks in MUC-2. Named-entity recognition for English is one of the tasks of MUC-6 which is organized in 1996. Training corpus is generated by annotating Wall Street Journal articles. ENAMEX (for people, organization, location) and NUMEX (time, currency, percentage) tags are introduced in this conference. 15 participants enrolled for the NER task and most of the results are successful with precision and recall values over 90%. Most successful system reached %97 precision, 96% recall values. (Grishman & Sundheim, 1996)

Cucerzan & Yarowsky (1999) is the first published NER research that includes Turkish. System is language independent and depends on bootstrapping algorithm with iterative learning on a character-based tree structure. System is built after the acceptance that words strongly tend to exhibit only one sense in a document. It uses a small named entity list about the source language as training seeds and morphological and contextual patterns as features. For example, “-escu” is stated as an almost perfect indicator for a last name in Romanian. This study reports 60% precision, 47% recall and 53% f-measure for Turkish.

Alfonseca & Manandhar (2002) built a general named entity recognition (GNER) system to find the most accurate generalization (hypernym) for an unknown concept or instance, by using WordNet ontology (lexical database). To classify an unknown instance, system runs queries on search engines to derive similarity scores for candidate words. Used notion here is that words semantically related must co-occur with the same kinds of words. This research extends the scope of NER with a more complex taxonomy structure. Domain specific documents are taken from the electronic version of “The Lord of the Rings” for experiments. Figure 3.3 shows the ontology

used and the classification results for concepts *hobbit*, *Mordor*, *Isengard*, *Hobbiton*, *wizard*, *horse* and *eagle*.

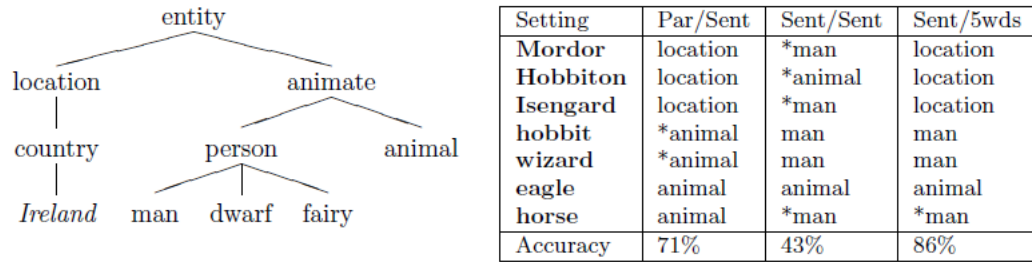


Figure 3.3 Ontology used in experiments and classification results (detected hypernyms) for 7-unknown concepts

Tür et al. (2003) developed an information extraction system for Turkish and NER is one of the tasks they worked on (other two are sentence and topic segmentation). Their NER approach is based on n-gram language models embedded in HMM. The study consists of four models: Lexical model uses boundary flags between word tokens to indicate name entity borders with *yes*, *no* and *mid* flags. Contextual model is used to capture information from surrounding context of word tokens. Morphological model uses case information (initial-upper, all-lower, all-upper, mixed etc.) alongside with a proper name database that stores common Turkish person, location and organization names. Tag model is only concerned with trigram possibilities for name entity tag (person, location, organization, else) and boundary flag (yes, no, mid) combinations. Newspaper articles are used for experiments. When all models combined, system has a success rate with 90.4% NE text accuracy, 92.73% NE type accuracy and 91.56% f-measure. Table 3.9 shows the NE tag probabilities calculated on contextual model for an unknown word following the word “Dr.” and a word boundary. *Person* appears to be the most suitable NE tag with 99% probability value.

Table 3.9 Example usage of contextual model for unknown words

Output Sequence	Probability
Dr./else boundary/yes unk/person	0.990119
Dr./else boundary/yes unk/location	0.000690
Dr./else boundary/yes unk/organization	0.000880
Dr./else boundary/yes unk/else	0002688

Like MUC, CoNLL events also give shared tasks about computational linguistics to participants and evaluate final results. Task in CoNLL-2003 is to build a language independent NER (English and German are the test languages); with a special challenge which is to include unannotated data to the training phase of the system. Participants are provided with different features (pos tag, chunk tags, affix information, gazetteers etc.) and given freedom to decide among them. It is observed that most participants used unannotated data to find out additional gazetteer terms. Interestingly enough, generally using only gazetteers seemed to provide more error reduction than systems that tried to find additional terms. On the other hand, using unannotated data to obtain capitalization information seemed to have positive effect on results (Sang & Meulder, 2003).

Wentland, Knopp, Silberer & Hartung (2008) built a multilingual named entity resource called HeiNER. Wikipedia is used as the main resource, as it contains a large amount of NEs compared to other commonly used lexical resources like WordNet. English is selected as the source language and Wikipedia cross-language links are used to build a translation dictionary to convert detected NEs to target languages. System also builds a disambiguation dictionary using redirect pages (for example “USA” and “United States of America” points to the same link) and disambiguation pages (for example term “Python” may indicate “Monty Python” or “Python” (programming language)). Another advantage of using Wikipedia articles is that, there is a high probability for an article heading to describe a NE. This surpasses some of the common NER problems like NE boundary detection or necessity of morphological normalization.

Küçük & Yazıcı (2009a) built a rule-based NER system for Turkish and tested its success on different domains (news articles, child stories, history texts). System uses lexical resources like dictionary of Turkish person names, list of well-known political people, list of well-known organizations and pre-defined pattern bases to detect possible NEs. Resulted f-measure is 78% for news articles domain; but it drops down to 69% for child stories and 55% for historical texts. Existence of foreign person names in child stories and absence of historical person and organization names in lexical

resources are determined to be leading causes for performance drops Results are in line with the general opinion that performance decrease is possible when rule-based NER systems are ported to other domains. Figure 3.4 shows the system's information source schema.

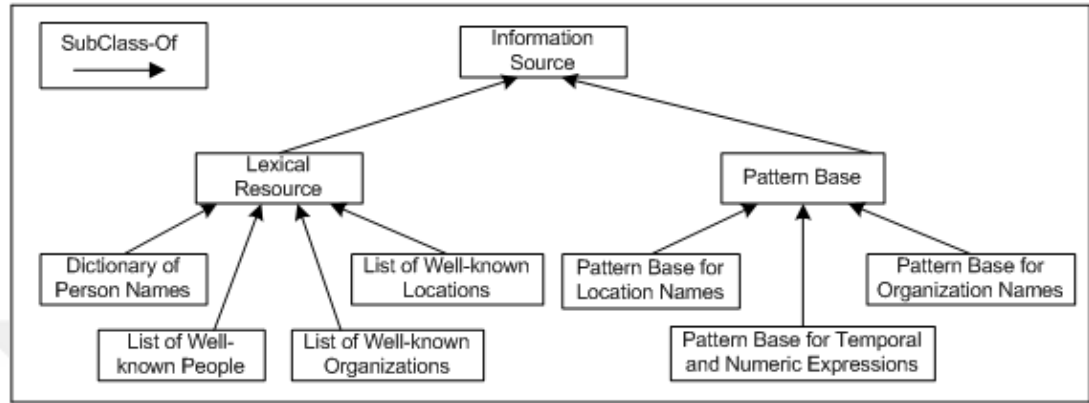


Figure 3.4 Information sources used for rule-based NER system

Küçük & Yazıcı (2009b) also tested their system on transcription test derived from video texts. 16 news videos from Turkish Radio and Television Company (TRT) archive are selected for experiments. Videos are manually transcribed as no automatic speech recognizer exists for Turkish back then. Evaluation resulted in a precision of 73%, recall of 77% and f-measure of 75%.

Tatar & Çiçekli (2011) described an automatic rule learning method using supervised learning. System starts with a set of named entities collected from a training dataset and generates rules from them. Main goal here is to get through domain adaptability problems, which is common for rule-based systems. System utilized from orthographical, contextual, lexical and morphological features. 2-level gazetteer structures are used in lexical model. For example, location is a higher level, more general categorization while location.country, location.city are secondary level, more specific classification. System is tested on Turkish news articles (TurkIE dataset) and resulted in a precision of 91.7%, recall of 90% f-measure of 91%.

Küçük & Yazıcı (2012) moved through their rule-based model and developed a hybrid system. 2 statistical features n (denotes the number of occurrences of an entity

text) and p (denoted the number of occurrences which happen to be annotated) are defined and p/n is used as a confidence value for each entity. In training phase, entities with high confidence values are extracted and added to the resources of recognizer. Significant performance improvement over rule-based system is observed with f-measure values of 85.9% on news data set, 85% on child stories and 66.9% on historical texts.

Şeker & Eryigit (2012) used conditional random fields (CRF) as their statistical model. Alongside with gazetteers, they used generator gazetteers (22 person, 44 location, 60 organization) that holds tokens that could come after or before regular words and construct NEs. 14 features are defined in 3 categories (morphological, lexical, gazetteer lookup). Windows width for CRF features is defined as $\{-3,+3\}$ where 0 is current token, +1 is next, -1 is previous token etc. Features are tested by including them one by one to the system. Experiments showed that all features but SS (start of sentence) had improved performance of the system. When all features included, system had reached 94.6% final f-measure in MUC metrics and 91.9% final f-measure in CoNLL metrics. Table 3.10 gives information about tested features and Table 3.11 shows the experiment results in MUC TYPE, MUC TEXT and MUC metrics.

Table 3.10 Tested feature description

Feature Code	Category	Description
STEM	Morphological	Stem information of the word's surface form
POS	Morphological	Part-of-speech tag information
NCS	Morphological	Noun case information (nominal or non-nominal)
PROP	Morphological	Proper noun information
INF	Morphological	All inflectional tags from morphological analysis
CS	Lexical	Case feature (lowercase, uppercase, proper name, mixed)
SS	Lexical	Start of the sentence information (is beginning or not)
BG	Gazetteer lookup	Indicates if token exists in a base gazetteer or not
GG	Gazetteer lookup	Indicates if token exists in generator gazetteer or not

Table 3.11 F-measure in MUC metrics on feature related experiments

Feature	MUC TYPE			Overall	MUC TEXT	MUC
	PER	ORG	LOC			
BM_stem	85.31	79.89	86.87	84.03	83.95	83.54
BM_surf	83.83	82.71	86.67	84.19	85.81	85.00
+STEM	85.62	83.26	87.26	85.30	87.08	86.19
+POS	87.34	83.08	87.47	86.06	88.11	87.09
+NCS	87.46	83.95	87.27	86.33	88.85	87.59
+PROP	88.87	85.12	88.68	87.68	90.98	89.33
+INF	89.65	85.32	89.79	88.38	91.60	89.99
+CS	92.76	89.09	89.85	90.92	94.73	92.83
+SS	92.75	89.01	90.15	90.97	94.68	92.83
+BG	94.00	89.82	92.20	92.27	95.50	93.89
+GG	94.81	91.09	93.35	93.29	95.89	94.59

Küçük, Jacquet & Steinberger (2014) performed NER experiments on Turkish tweets. 2320 tweets are collected to form data set. Besides seven basic types (person, location, organization [these three are also called as PLO], date, time, money, percent), a misc type (product names, tv shows, music bands etc.) is also used for annotation. Hashtag usage is also suggestive as it is common to have NEs in hashtags. Two lists for person and organization names, which are detected to be used as single tokens in news articles (at least 30 times in Europe Media Monitor database) are built and used in system. Results show that 25% of PLO initial letters are not properly capitalized, only 32% of person names are composed of first name-surname pairs and % 10 of PLO text has affected from normalization of Turkish characters. Another problem is the multiword NE tokens in hashtags that are written without whitespace. System reached 66% precision, 31.5% recall and 42.6% f-measure values.

Küçük, Küçük & Arıcı (2016) composed and shared a dataset comprising news articles in Turkish with named entities annotated, for general use of NER studies. 10 news articles from METU Turkish Corpus are selected and final annotation document consists of 1425 named entities (398 person, 567 location, 460 organization).

Şeker & Eryiğit (2016) moved through their study in 2012 and added TIMEX and NUMEX entity types. They also worked on a new dataset (Web2.0 domain) with user generated content (UGC). Additional features like numeric value, percentage sign etc. are defined and used for new entity types. A lexicon named Auto Capitalization

Gazetteer (CAP) is constructed, which contains gazetteer terms that are unlikely to be used as common noun. Unlike their previous study, this time feature performances are tested by removing them from the complete model one by one. This way SS (start of sentence) feature is determined have 2.11% positive effect on performance. Experiments on UGC data set resulted with 67.9% success on best model. When CAP feature is removed it causes more than 20% performance loss.

Sahin, Tirkaz, Yildiz, Eren & Sonmez (2017) automatically classified Turkish Wikipedia pages to construct a corpus for NER task. Constructed corpus contains approximately 300K entities. Entities are categorized in 77 different domains to provide fine-grained classification, and those domains are grouped in four coarse-grained types (person, location, organization, misc). A semantic knowledge base named Freebase is used to overcome noisy data and ambiguities on text. Highest f-measure is calculated as 84% for the system.

Güneş & Tantuğ (2018) proposed a bidirectional long short-term memory neural network structure and tested it on six different models. When base input set that only contains word vectors is used, highest f-measure value is calculated as 91.59%. When orthographical and morphological attributes are included in input set and used on a multilayer neural network model, system reached 93.69% f-measure value.

Güngör, Üsküdarlı & Güngör (2018) proposed a neural network model for Turkish NER task, which creates a context vector for every position in the sentence by processing the words in forward and backward directions. It is aimed to detect inner-word relations with these vectors as they provide character level information unlike distributional word vectors. Success of the system, which is calculated as 90.96% f-measure when only distributional word vectors are used, has shown improvement with 93.37% f-measure when character level word vectors are included.

3.5 Exam and Question Generation

Studies about exam and question generation is generally centered around two approaches: Effectively using large scale question banks containing categorized

questions and using question templates with numeric parameters to produce dynamic questions, mostly for math and science subjects. Some of the remarkable studies are mentioned on this section. On the other hand, generating exam questions using the context of an input document on time, which is aimed to accomplish by the proposed educational software, is not a well-studied area with very few studies.

Baklavas, Economides & Roumeliotis (1999) compared web-based test tools with respect to the supported question type variety, multimedia usage, security, easiness of development, maintenance and delivery of tests and automatic grading and analysis of results. Test questions are created by instructors via on-screen instructions or selected from a question bank on these tools. Cyber Exam and QuestionMark Perception are stated as best choices based on the criteria and practical experience.

Instead of choosing from an existing set of problems, Lee (2000) proposed using a set of templates to generate different variables for the same question. System is tested on 120 test questions (each with numeric variables over graphics), from The Fundamentals of Engineering (FE) examination using over 500 templates. The generator program changes problem variables, correct answers, wrong answers (in a reasonable range) and order of exam questions. Figure 3.5 shows an example testing page with a generated question.

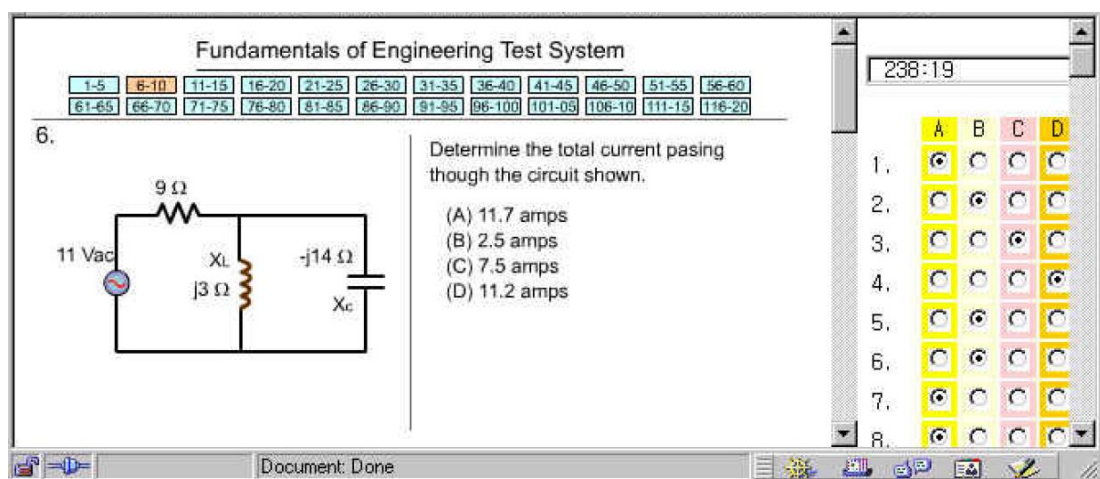


Figure 3.5 The exam testing page with an example generated question

DePiero (2001) proposed NetExam, a web-based testing engine, which generates exams on demand by randomly selecting questions from given subject categories from a database. Study also worked on the assessment process after a group of students completed an exam, with computed statistics over submitted and automatically graded exams and the provided bulletin board style comments section.

Shende, Dalch & Warner (2002) projected to unite exam administrators, examinees, question providers and question approvers on an exam distribution server with inner modules like exam generation, exam question approval, exam scoring etc. Proposed model benefits from a database of exam questions grouped in predetermined sections. To collect meaningful questions, system is designed not to add suggested questions to database before an approval process.

Using R programming, Grun & Zeileis (2009) developed the package exams, to provide a framework for automatic generation of statistical exams. System exploits a pool of exercises and a master file to identify the layout of the final PDF document. Solutions of the provided types of exercises are either multiple-choice answers, numeric values and short text answers. A separate Sweave file, which contains R code for data generation and solution calculation, and LaTeX code to store metadata about question and solution environment, is used to specify each exercise in the pool.

Ugurdag et al. (2009) worked on converting a static multiple choice math/science question (with constant numbers and choices) to a dynamic question using parameters and proposed the concept smart question (sQ) in line with this purpose. A parameter is used to express the initial numbers given in a question. A smart question generation starts with assigning an unaltered image file (the original question) and continues with parameter definition, parameter specification and question generation phases. Using this concept, a range of values or a formula is specified for each parameter and variety in answer set is provided.

Özkul (2009) presented a technique to automate test preparation of quiz questions, answer keys and student feedback. The project mainly aims to standardize questions

and their numerical answer set. For this purpose, past quiz questions written in Microsoft Word and answer keys in Microsoft Excel are used to create templates. Afterwards, problem generator is executed to generate random numbers within defined limits to be used on templates. It is stated that the system has been used at Operations Management course at the State University of New York (SUNY) College successfully since 2004 for eleven quizzes including 22 questions.

Liu, Shi, Liu & Li (2010) focused on to enhance question selection from a question bank for an automatic test generation process and proposed a multi-constrained model based on genetic algorithm. Constraints defined for a test paper include score, answer time, questions forms, difficulty, chapters and teaching requirement. Out of 7-dimensional variable space, difficulty, chapters and teaching requirement are selected to construct a 3-dimensional model to achieve a test paper that meets the user's needs.

Hussein, Elmogy & Guirguis (2014) proposed a model for English question generation task that selects one sentence at a time, extracts sections of it and uses patterns and transformation rules to construct a question. System uses OpenNLP statistical parser for training purposes, mainly to tokenize sentences into phrases and detect part-of-speech (POS) tags for these phrases. Template rules are scanned to find a suitable question phrase (what, where, who, when, how much) for tagged phrases. Users are also allowed for modification (which adds a new template rule to database), as they can edit generated question text, choose a level of hardness and change the question phrase. Table 3.12 shows some of the possible questions derived from a sentence.

Table 3.12 Possible questions derived from example sentences

Given Statement	Possible Question
Ali is going to London.	Where is Ali going to?
	Who is going to London?
Ahmet plays football in Egypt team.	Where does Ahmet play football?
	Who play football in Egypt team?
Ali played football in Cairo.	Where did Ali play football?
	Who play football in Cairo?
I found my books on the table.	Unmatched (New template rule needed)

CHAPTER FOUR

COURSE DOCUMENT CLASSIFICATION

4.1 Overview and Dataset Introduction

In the earlier phases of the project, user is expected to indicate the course of the provided lecture note (as History or Geography) to the system. This is thought to be an improvable approach later, with the automatic classification of the documents. Aim here is to find out effective ways to auto-identify the class label of Turkish lectures notes by comparing many aspects of a complete document classification process with many experiments. CGR (for Geography) and TRH (for History) are the specified document labels. 1200 proper documents (600 for CGR and 600 for TRH) are collected from various publicly available web-based or written sources to construct a lecture notes dataset to be used on experimentation phase. This is the only module in this thesis that uses a bag-of-words approach, in which occurrence and frequency of words within a document is the main concern and word order is disregarded. Therefore it doesn't require an initial sentence boundary detection service.

4.2 Compared Approaches

Four aspects of a document classification task specialized for geography and history domains are compared to find out the most suitable model to be used on exam module.

4.2.1 Existence of Stop Words

The most common words in a language, which have very little meaning (like “a, an, the, on, in, at” in English) are called as stop words. Dataset is used both with (SW) and without Turkish stop words (No-SW) to see how presence – absence of them effects the results. The stop word list of an open source project name “TrStop” is used for this assignment (Aksoy & Öztürk, 2016). This is a comprehensive and up-to-date source for this purpose with a total of 285 words. 278 of them are selected for usage and stored as a text document. Complete stop word list is given in Appendix-1.

4.2.2 Stemming Approaches

Stemming in agglutinative languages refers to a heuristic process that aims to reach to the stem (or root in some approaches) of a word by extracting inflectional suffixes (also derivational suffixes in some approaches), instead of using its surface form. Five approaches are compared on this module, which are No Stemming (keeping words in their surface forms) (No-S), F4 (truncating words after 4 characters), F5, F6 and Zemberek Stemmer (ZS).

4.2.3 Classification Algorithms

Classification is taken as a machine learning task to be handled with supervised learning approach in this thesis. Accordingly, seven different classification algorithms are compared here. They are Naïve Bayes Multinomial (NB-M), Naïve Bayes Bernoulli (NB-B), k-NN (k-Nearest Neighbor) by taking k as 1,3 and 5 separately, Decision Trees (DT) and Support Vector Machines (SVM).

4.2.4 Feature Selection Methods

Feature selection is the process of selecting a subset of features to be used in model construction, based on their scores in statistical tests for their correlation with the outcome variable, instead of using all features. In document classification, to reduce the number of input words to be used for analysis and find most distinguishing ones, feature selection can be applied. Four different approaches are compared here, which are No Feature Selection (No-FS), Information Gain (IG) with ranker, Correlation Coefficient (CC) and taking the most frequent 500 words (MF-500) for each class.

4.3 Experimentation Phase

Besides the compared approaches, using different sizes for different – test data set is another observation. First, complete data set is split as 10% training and 90% test, then 50% training and 50% test. This makes a total of 560 experiments. Weka is the used platform to perform experiments, after preprocessing of collected raw documents.

4.3.1 Document Preprocessing

Preprocessing phase includes punctuation and symbol removal, excessive whitespace corrections, lowercase transformation, extraction of single digit numbers and letters. It is observed that some of the raw documents also had control characters which are irrelevant and also removed.

If it's a requirement for the particular experiment, stop word elimination and the related stemming operation are also applied to put the text into its final form. Figure 4.1 shows an example output from a three staged preprocessing (text normalization, stop word removal and stemming using Zemberek respectively) applied on a raw document text.

1 - Raw Text	2 - After Normalization	3 - After Stop Word Removal
Savaşın Sona Ermesi I. Dünya Savaşı'nın daha birinci yılında her iki blok da birbirine karşı üstünlük sağlayamadı. Bu nedenle savaş, büyük ölçüde bir siper savaşı niteliğine bürünmüştü. 1915 yılında Bulgaristan'ın Bağlaşma Bloğu'nda, İtalya'nın ise, Anlaşma Bloğu'nda savaşa girmesi de dengeyi değiştiremedi. 1916'da savaşın Anlaşma Devletleri lehine gittiğini düşünen Romanya, Anlaşma devletleri yanında savaşa girdi. Buna	savaşın sona ermesi i dünya savaşının daha birinci yılında her iki blok da birbirine karşı üstünlük sağlayamadı bu nedenle savaş büyük ölçüde bir siper savaşı niteliğine bürünmüştü 1915 yılında bulgaristanın bağlaşma bloğunda italyanın ise anlaşma bloğunda savaşa girmesi de dengeyi değiştiremedi 1916da savaşın anlaşma devletleri lehine gittiğini düşünen romanya anlaşma devletleri yanında savaşa girdi buna rağmen güç durumunda kaldı ve 1917de yenildi yunanistan ise venizelosun çıkardığı ayaklanma sonrasında haziran 1917de anlaşma	savaşın sona ermesi dünya savaşının birinci yılında blok birbirine karşı üstünlük sağlayamadı savaş büyük ölçüde siper savaşı niteliğine bürünmüştü 1915 yılında bulgaristanın bağlaşma bloğunda italyanın anlaşma bloğunda savaşa girmesi dengeyi değiştiremedi 1916da savaşın anlaşma devletleri lehine gittiğini düşünen romanya anlaşma devletleri yanında savaşa girdi güç durumda kaldı 1917de yenildi yunanistan venizelosun çıkardığı ayaklanma sonrasında haziran 1917de anlaşma devletlerine katıldı 1917 ekim devrimi rusyanın savaştan çekilmesi
4 - After Stemming		
savaş son er dünya savaş birinci yıl blok birbir karşı üstün sağla savaş büyük ölçü siper savaş nitelik bürü yıl bağlaş anlaşma savaş gir denge değiş da savaş anlaşma devlet leh git düşün anlaşma devlet yan savaş girdi güç durum kal de yen çıkar ayakla sonra haziran de anlaşma devlet katıl ekim devrim savaş çekil bağlaş devlet üstün konum gel etki doğu cephe kapan durum bağlaş devlet savaş kazan yeter değil kara ordu üstün bağlaş devlet geçmiş deniz üstün anlaşma devlet i özel deniz tartışma üstün devam et üstelik sömürge devlet büyük insan güç sağla savaş sömürge kaybet insan güç sınır bu birlikte devlet ekonomik askeri bakım muhtaç i ekonomi savaş kilitle uza çıkmaz girdi üzeri dan denizaltı savaş başvuru denizaltı anlaşma blok asker malzeme taşı gemi batır başla dünya savaş baş taraf ilan abd özel firma		

Figure 4.1 Example output on different preprocessing stages of a raw document text

Afterwards, the prepared text is used to create UTF-8 encoded XML formatted text files with 3 tags (Course, Title, Text + “_StemmingApproachUsed”) to derive a standardized format. Example standardized documents derived from the same history lecture note after preprocessing phase are shown on following two figures. Figure 4.2 represents the output when No Stemming approach is applied, and Figure 4.3 represents the output when F5 stemming approach is applied. Stop words are removed on both documents.

```

<COURSE>TRH</COURSE>
<TITLE>Halkçılık</TITLE>
<TEXT_NO_STEMMING>halkçılık devlet yönetiminde halka dayanma halktan güç
alma halkın egemenliğine sığınma özellikleri içerir halkçılık ilkesine
kişiye aileye zümreye sınıfa imtiyaz tanınmaz milletin fertleri kanun
önünde eşittir sınıf egemenliğini reddeder bireyler arasında alanda
fırsat eşitliğini amaçlar milli egemenliği esas alır halkın yönetmesini
demokrasiyi öngörür halkçılığın amacı halkın refah mutluluğunu
sağlamaktır halkın ülke kaynaklarından eşit yararlanmasını sağlar
halkçılık devrimler aşar vergisinin kaldırılması türk medeni yasasının
kabulü kılık kıyafet kanununun kabulü kadınlara siyasi hakların verilmesi
soyadı kanununun kabulü not halkçılık cumhuriyetçilik milliyetçilik
ilkelerinin doğal sonucudur not halkçılık anayasada türkiye cumhuriyeti
sosyal hukuk devletidir şeklinde yansımıştır</TEXT_NO_STEMMING>

```

Figure 4.2 Standardized document example with No Stop Words and No Stemming approach

```

<COURSE>TRH</COURSE>
<TITLE>Halkçılık</TITLE>
<TEXT_F5>halkç devle yönet halka dayan halkt güç alma halkı egeme sığın
özelli içerii halkç ilkes kişiy ailey zümre sınıf imtiy tanın mille fertl
kanun önünd eşitt sınıf egeme redde birey arası aland fırsa eşitl amaçl
milli egeme esas alır halkı yönet demok öngör halkç amacı halkı refah
mutlu sağla halkı ülke kayna eşit yarar sağla halkç devri aşar vergi
kaldı türk meden yasa kabul kılık kıyaf kanun kabul kadın siyas hakla
veril soyad kanun kabul not halkç cumhu milli ilkel doğal sonuc not halkç
anaya türki cumhu sosya hukuk devle şekli yansı</TEXT_F5>

```

Figure 4.3 Standardized document example with No Stop Words and F5 stemming approach

4.3.2 ARFF File Generation

ARFF (Attribute-Relation File Format) is the required file format to carry out machine learning experiments on Weka. An ARFF file is a text file that describes the instances sharing a set of attributes. ARFF files are composed of two sections: Header section contains the name of the relation, a list of the attributes and their types, while data section contains the data instances denoted with the defined attributes. While data mining tasks with numeric values often exploit high numbers of attributes, ARFF files used for this task contains 2 attributes for each document instance as text (string typed) and class (nominal typed with possible values {CGR, TRH}).

Weka allows using separate ARFF files for training and test data or using a single ARFF file with combined data. If second option is selected, proportion of training to test data has to be specified before experiments. In this task, preprocessed and

standardized lecture notes are gathered up to obtain a single ARFF file for each five stemming approaches, with or without stop words in text, which makes a total of 10 files. Thus, each ARFF file contains a total of 1200 data instances to denote every lecture note with its class label. Words of document instances collected in ARFF files are used to derive feature vectors for experiments. Every ARFF file is used to perform 28 experiments (for 50% training – 50% test distribution), as selection from among seven classification algorithms and four feature selection methods are made on Weka interface. Figure 4.4 shows header section and first two instances from data section of an example ARFF file that is generated for F6 stemming with no stop word removal approaches.

```
@relation 'F6_StopWordNotEliminated'

@attribute text string
@attribute @@class@@ {CGR,TRH}

@data
'21 aralık yıllık hareke sırası dünyan 21 aralık günü geldiğ konumd güney
yarım küre güneşe dönükt bu durumd güneş ışınlı öğle vakti ekvato 23 27
güneyi öğlak dönenc dik açıyla düşer bu tarih güney yarım küre için yaz
mevsim başlan yani yaz gün dönümü solsti dır kuzey yarım kürede ise kış
mevsim başlan yani kış gün dönümü solsti dır güney yarım küre yaz başlan
en uzun gündüz yaşanı güney kutup daires 24 saat gündüz yaşanı güney kutup
çember tamame aydınl güneş ışınlı en büyük açıyla düşer öğle vakti yıl
içinde en kısa gölge oluşur kuzey yarım küre kış başlan en uzun gece
yaşanı kuzey kutup daires 24 saat gece yaşanı kuzey kutup çember tamame
karanlı güneş ışınlı en küçük açıyla düşer öğle vakti yıl içinde en uzun
gölge oluşur',CGR
'21 hazira yıllık hareke sırası dünyan 21 hazira günü geldiğ konumd kuzey
yarım küre güneşe dönükt bu durumd güneş ışınlı öğle vakti ekvato 23 27
kuzeyi yengeç dönenc dik açıyla düşer bu tarih kuzey yarım küre için yaz
mevsim başlan yani yaz gün dönümü solsti dır güney yarım kürede ise kış
mevsim başlan yani kış gün dönümü kuzey yarım küre yaz başlan en uzun
gündüz yaşanı kuzey kutup daires 24 saat gündüz yaşanı kuzey kutup çember
tamame aydınl güneş ışınlı en büyük açıyla düşer öğle vakti yıl içinde en
kısa gölge oluşur güney yarım küre kış başlan en uzun gece yaşanı güney
kutup daires 24 saat gece yaşanı güney kutup çember tamame karanlı güneş
ışınlı en küçük açıyla düşer öğle vakti yıl içinde en uzun gölge oluşur',CGR
```

Figure 4.4 ARFF file example for F6 stemming and no stop word removal approaches

4.3.3 Interpretation of Experiment Results

Precision (fraction of relevant instances among the retrieved instances), recall (fraction of relevant instances that have been retrieved over the total amount of relevant instances), f-measure (accuracy measure that takes harmonic mean of

precision and recall) and incorrectly classified instance percentage are the metrics used for comparisons. First, only 10% of the complete dataset is used for training (120 documents) and 90% is used for test (1020 documents) to check whether any combination of approaches yields satisfactory results. Table 4.1 shows the results in f-measure, with best scores highlighted.

Table 4.1 Course document classification experiment results (10% training – 90% test)

F - MEASURE (10% Training - 90% Test)		SW					No-SW				
		No-S	F4	F5	F6	ZS	No-S	F4	F5	F6	ZS
NB-M	No-FS	0.96	0.95	0.95	0.95	0.96	0.96	0.95	0.95	0.95	0.95
	IG	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.96
	CC	0.96	0.96	0.96	0.97	0.96	0.96	0.96	0.97	0.97	0.96
	MF-500	0.96	0.96	0.96	0.95	0.95	0.96	0.96	0.96	0.95	0.95
NB-B	No-FS	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96
	IG	0.96	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.96	0.97
	CC	0.96	0.97	0.97	0.97	0.97	0.96	0.97	0.97	0.97	0.96
	MF-500	0.96	0.97	0.96	0.96	0.97	0.96	0.96	0.96	0.96	0.96
1-NN	No-FS	0.38	0.45	0.40	0.39	0.81	0.38	0.41	0.38	0.41	0.75
	IG	0.43	0.46	0.47	0.43	0.51	0.43	0.45	0.46	0.42	0.49
	CC	0.42	0.46	0.45	0.40	0.50	0.40	0.45	0.45	0.40	0.49
	MF-500	0.64	0.65	0.68	0.63	0.67	0.61	0.63	0.62	0.59	0.62
3-NN	No-FS	0.36	0.42	0.40	0.38	0.81	0.35	0.37	0.39	0.37	0.58
	IG	0.38	0.40	0.42	0.39	0.44	0.37	0.39	0.40	0.38	0.43
	CC	0.37	0.41	0.41	0.39	0.44	0.36	0.40	0.40	0.37	0.42
	MF-500	0.59	0.60	0.61	0.57	0.61	0.60	0.59	0.57	0.52	0.56
5-NN	No-FS	0.35	0.39	0.39	0.37	0.75	0.35	0.39	0.38	0.36	0.60
	IG	0.36	0.38	0.40	0.37	0.41	0.36	0.38	0.39	0.37	0.40
	CC	0.36	0.39	0.38	0.37	0.43	0.35	0.38	0.39	0.36	0.42
	MF-500	0.56	0.58	0.56	0.54	0.61	0.54	0.57	0.53	0.51	0.55
DT (J48)	No-FS	0.79	0.84	0.88	0.83	0.84	0.76	0.84	0.87	0.83	0.84
	IG	0.85	0.88	0.89	0.88	0.84	0.85	0.88	0.90	0.88	0.84
	CC	0.85	0.88	0.90	0.88	0.84	0.82	0.88	0.89	0.88	0.84
	MF-500	0.76	0.86	0.88	0.83	0.87	0.76	0.87	0.89	0.83	0.87
SVM	No-FS	0.94	0.92	0.93	0.93	0.94	0.93	0.92	0.93	0.93	0.94
	IG	0.92	0.92	0.94	0.93	0.94	0.92	0.92	0.93	0.94	0.94
	CC	0.94	0.94	0.93	0.93	0.95	0.93	0.94	0.93	0.93	0.93
	MF-500	0.94	0.94	0.94	0.93	0.94	0.95	0.94	0.94	0.93	0.93

Highest f-measure value is observed as 97% and 20 different combinations of approaches have reached that peak score. All of these 20 experiments are executed under NB-M or NB-B algorithms, used a stemming approach and used a feature selection method. Other observed results are listed as follows:

- NB-B and NB-M are capable to classify the instances very accurately using only 10% training data.
- NB-B's overall performance is slightly better than NB-M. 16 out of 20 most successful experiments are performed with this algorithm.
- SVM results are also satisfactory in general, as it never dropped below 90% in none of the experiments. Generally, SVM performance is increased with ZS.
- DT algorithm reached its highest success with F5 stemming approach.
- Using ZS alongside with No-FS makes a huge impact on kNN algorithm's f-measure values. Using MF-500 as feature selection approach also has a good effect on kNN algorithm success.
- But within the scope of this dataset, kNN can't be considered as an effective classification algorithm, as its lowest incorrectly classified instances score is 17.96% (with k=3, SW, No-FS and ZS as stemming approach) and generally this score is resulted to be around 40%.
- In most of the cases, using a feature selection method seems to have a good impact on general success rate.
- Among the 10 stemming and stop word existence approach combinations, there are no significant performance differences but ZS with SW seems to be a slightly better pair.
- Among the 28 classification algorithm and feature selection method combinations, using NB-B with IG has the best success rate.
- In most cases, NSW approach didn't make a good impact.
- When F4, F5 or F6 stemming is performed, 1-NN gives better results than 3-NN and 5-NN.
- Using a stemmer gave better results rather than No-S approach for most cases.
- Comparing F4, F5 and F6 stemmers is not a very feasible task with close f-measure scores.
- Half of the 20 most successful combinations use CC as feature selection method, while 8 of them use IG and 2 of them use MF-500.

When the small training size is considered, some remarkable experiment results are observed. However, their sufficiency is still questionable to be used on examination

module. So experiments are repeated with increased training set size as 50% of the complete dataset is used for training and 50% is used for test (600 documents for each). Table 4.2 shows the results in f-measure, with best scores highlighted.

Table 4.2 Course document classification experiment results (50% training – 50% test)

F - MEASURE (50% Training - 50% Test)		SW					No-SW				
		No-S	F4	F5	F6	ZS	No-S	F4	F5	F6	ZS
NB-M	No-FS	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
	IG	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.99
	CC	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.99
	MF-500	0.97	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98
NB-B	No-FS	0.96	0.96	0.97	0.97	0.96	0.97	0.96	0.97	0.97	0.97
	IG	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	CC	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97
	MF-500	0.96	0.96	0.97	0.96	0.97	0.96	0.96	0.97	0.96	0.96
1-NN	No-FS	0.46	0.55	0.54	0.48	0.57	0.43	0.53	0.51	0.48	0.55
	IG	0.58	0.63	0.62	0.57	0.65	0.57	0.64	0.61	0.57	0.65
	CC	0.57	0.63	0.59	0.55	0.65	0.53	0.60	0.58	0.54	0.67
	MF-500	0.74	0.77	0.80	0.77	0.81	0.72	0.73	0.78	0.76	0.77
3-NN	No-FS	0.42	0.49	0.44	0.43	0.49	0.40	0.44	0.42	0.41	0.43
	IG	0.50	0.54	0.54	0.50	0.58	0.47	0.54	0.51	0.48	0.56
	CC	0.49	0.55	0.53	0.46	0.60	0.44	0.54	0.52	0.46	0.58
	MF-500	0.70	0.74	0.75	0.70	0.78	0.67	0.69	0.73	0.70	0.72
5-NN	No-FS	0.40	0.46	0.42	0.41	0.44	0.37	0.41	0.41	0.39	0.41
	IG	0.46	0.50	0.49	0.45	0.54	0.44	0.50	0.48	0.44	0.51
	CC	0.45	0.51	0.48	0.44	0.55	0.41	0.49	0.47	0.43	0.53
	MF-500	0.68	0.71	0.73	0.68	0.77	0.64	0.66	0.69	0.66	0.69
DT (J48)	No-FS	0.87	0.88	0.91	0.91	0.92	0.88	0.88	0.91	0.91	0.93
	IG	0.89	0.89	0.90	0.92	0.92	0.89	0.89	0.90	0.92	0.92
	CC	0.89	0.89	0.90	0.92	0.91	0.89	0.89	0.91	0.92	0.92
	MF-500	0.88	0.88	0.91	0.91	0.90	0.89	0.88	0.91	0.90	0.93
SVM	No-FS	0.96	0.95	0.96	0.95	0.96	0.96	0.94	0.97	0.96	0.95
	IG	0.96	0.95	0.96	0.96	0.97	0.95	0.95	0.96	0.96	0.97
	CC	0.95	0.95	0.95	0.94	0.97	0.96	0.95	0.94	0.95	0.97
	MF-500	0.95	0.95	0.95	0.96	0.96	0.96	0.95	0.95	0.96	0.96

Highest f-measure value is observed as 99% and 3 different combinations of approaches have reached that peak score. All of these 3 experiments are executed under NB-M algorithm, used ZS as stemming approach and used a feature selection method. Other observed results are listed as follows:

- After the training portion is increased, NB-M classification algorithm took the lead from NB-B with slightly more accurate results.
- SVM also caught up with NB-B. These three algorithms are still the most reliable selections.
- DT algorithm is not far behind them with passing 90% f-measure in many experiments.
- No consistent positive impact is observed from NSW approach.
- Using MF-500 as feature selection approach has a good effect on kNN algorithm success.
- Using a stemmer increases the performance for most of the cases.
- Again, using ZS with SW has the best performance among all stemming and stop word existence approach combinations.
- This time, using NB-M with IG has the best success rate among all classification algorithm and feature selection method combinations. Actually, all combinations with NB-M turned out to be top four approaches within 28.
- All 10 most successful experiments are executed under NB-M algorithm and nine of them used a feature selection method (IG or CC).
- All 5 most successful experiments used ZS. Among the top 10, F5 is also used 3 times.

4.3.4 Selected Classification Model

Figure 4.5 shows the result screen of one of the most successful experiments (NB-B, F5, IG, NSW) when 10% of the dataset is used for training. Figure 4.6 shows the result screen of one of the most successful experiments (NB-M, ZS, CC, NSW) when 50% of the dataset is used for training. As expected, percentage of incorrectly classified instances is decreased from 2.68% (29 out of 1080) to 0.83% (5 out of 600) when training portion is increased. Therefore, classification model to be used on final educational software is selected from the second experiment set. Based on observations and used approaches on the most successful experiments, **NB-M** as classification algorithm, **ZS** as stemming approach, **IG** as feature selection method is selected. NSW approach had minor impact on this dataset but doing a stop word

elimination is mostly preferred as it reduces the amount of unrelated words. Also it is used on 2 of the 3 most successful combinations, so **NSW** is selected as the stop word existence approach.

```

=== Summary ===

Correctly Classified Instances      1051          97.3148 %
Incorrectly Classified Instances    29           2.6852 %
Kappa statistic                    0.9463
Mean absolute error                 0.0262
Root mean squared error             0.1587
Relative absolute error              5.2378 %
Root relative squared error         31.7353 %
Total Number of Instances          1080

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,978   0,031   0,969     0,978   0,973     0,946   0,994    0,992    CGR
                0,969   0,022   0,978     0,969   0,973     0,946   0,992    0,985    TRH
Weighted Avg.   0,973   0,027   0,973     0,973   0,973     0,946   0,993    0,988

=== Confusion Matrix ===

  a  b  <-- classified as
528 12 |  a = CGR
 17 523 |  b = TRH

```

Figure 4.5 Result screen of NB-B, F5, IG, NSW experiment (10% training – 90% test)

```

=== Summary ===

Correctly Classified Instances      595          99.1667 %
Incorrectly Classified Instances    5           0.8333 %
Kappa statistic                    0.9833
Mean absolute error                 0.0089
Root mean squared error             0.091
Relative absolute error              1.7847 %
Root relative squared error         18.1976 %
Total Number of Instances          600

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,990   0,007   0,993     0,990   0,992     0,983   1,000    1,000    CGR
                0,993   0,010   0,990     0,993   0,992     0,983   0,998    0,996    TRH
Weighted Avg.   0,992   0,008   0,992     0,992   0,992     0,983   0,999    0,998

=== Confusion Matrix ===

  a  b  <-- classified as
297  3 |  a = CGR
  2 298 |  b = TRH

```

Figure 4.6 Result screen of NB-M, ZS, CC, NSW experiment (50% training – 50% test)

Besides increasing classification accuracy by using a more relevant set of word features, using a feature selection method also has a decent effect on execution time. Experiments on the selected model are executed on **0.03 seconds**, while performance of the model with same classification algorithm, stemming approach and stop word existence choice, used with no feature selection method is measured as **0.07 seconds**.

Selected feature selection method IG assigns a score between 0 and 1 for each word feature to indicate how much information it gives about the classification decision. Highest scored features indicate the most distinctive words. To perform experiments, threshold value is selected as 0.015 to eliminate features with lower scores. 1226 word features passed this limit and remained to build classifier. 10 highest scored words are shown on Table 4.3. Extended list with the most distinctive 100 words is given in Appendix-2.

Table 4.3 Most distinctive 10 words for classification task based on their IG scores

Word	IG Score
devlet	0.52314
savaş	0.3528
karşı	0.208
fazla	0.20644
yağ	0.19711
sıcaklık	0.19198
ordu	0.18133
iklim	0.18096
iste	0.17603
birlik	0.17435

4.4 Classification Model Integration with Exam Module

Weka, the machine learning library used to perform course document classification experiments is originally developed for Java platforms. To enable its usage in .NET platform and integrate with exam module, system benefits from IKVM.NET project that provides .NET implementation of Java class libraries. Weka libraries are used to set classification algorithm (NB-M) and feature selection method (IG) properties and perform the classification job based on the input course document supplied by user.

To add external packages of compiled codes (DLL) about a specific task on a Microsoft Visual Studio project, using NuGet packages that support code sharing between developers is a common approach. As ZS is the selected stemming approach, the NuGet package named NZemberek which provides the required libraries to include Zemberek features to .NET platform is installed on the target project.

ARFF file is prepared dynamically by preparing and appending the input course document context at the end of training document instances. Final ARFF file contains 601 instances of documents, as 600 of them (300 with CGR, 300 with TRH labels) are used for training and the last one, which is expected to be classified successfully, is used for test.

Documents classification operation is started when user loads a text-based lecture note in Turkish to the system, on new test screen. This screen also contains the changeable exam constraints like question types to include and number of questions, so classification operation is expected not to lock other controls when executed. Multithreading approach is used to provide this, as document classification operation is treated as a separate task. Example screenshots on exam module use case scenarios, including new test screen after document classification execution are given under Chapter 8.

Information about every document that is loaded to the system and classified based on its domain is stored in database. This allows to prevent redundant document classification overhead, if recently loaded document is already analyzed and classified before. However, comparing filenames of documents can't guarantee a duplication single-handedly. So filenames along with number of sentences and number of headings are specified to be the distinctive property set for a document when comparisons are made. Database structure is explained in more detail in Chapter 8.

CHAPTER FIVE

SENTENCE BOUNDARY AND HEADING DETECTION

5.1 Overview

As most of the tasks within this thesis require sentence-level operations, sentence boundary detection (SBD) is a top priority and its accuracy directly effects the success rate of other tasks. Besides, SBD is not considered as a stand-alone task as developed module also works on detection of headings and itemized text parts in given course document. Proposed model is a rule-based approach that is based on pre-defined sentence boundary rules along with a range of controls about in-text conditions. Instead of directly applying SBD rules on raw input text, deriving headings and paragraphs is the initial sub-task here. Correlations between headings and paragraphs are also examined on this stage and a heading is assigned for each paragraph. SBD operations are applied on obtained paragraphs to derive sentences afterwards. Some of the pre-defined sentence boundary rules are shown on Table 5.1 (LC → lowercase character, UC → uppercase character, WS → whitespace, NWS → not whitespace, D → digit, true → indicates a sentence boundary condition, false → indicates a not sentence boundary condition, other characters are self-explanatory).

Table 5.1 Example pre-defined rules about sentence boundary conditions

Condition	Output
LC . UC	True
LC . D	True
LC . WS UC	True
LC . WS D	True
LC) . UC	True
LC ? UC	True
LC ! D	True
LC . “ NWS	True
LC . LC	False
UC . LC	False
LC . ” WS	False
LC : “ UC	False
D . D	False
(!)	False

5.2 Regular Expression Usage

A regular expression (RegExp) is a pattern used to match character combinations in strings. Rules defined for sentence boundary conditions are translated into a single RegExp on back-end side for system usage. Apart from that, SBD module benefits from different RegExps used for separate tasks. Table 5.2 shows the RegExps defined and what they are used for. They are enumerated for easier mention on later sections. (Note that given RegExps are used on .NET platform and minor changes might require for different working environments.)

Table 5.2 Regular expressions defined and used in SBD module

#	Regular expression details (Name – text – short description)
RegExp1	Sentence boundary rule (?<=[a-zıüğçşö][\])?\s*?[\.,! ?][\s]*)?(?=[A-Z0-9İÜÖÇŞ] - - ["'"][\s])
	Derived from pre-defined rules and used as main separator for SBD task.
RegExp2	Reference format (\s?)(\s?\w+, \s?\d{2,4})(\s?:\s?\d+)?(\s?)
	Used to indicate in-text reference format. Example → (Behar, 1996: 63)
RegExp3	Irrelevant separator format ^(- _ * #){5,}
	Used to detect and remove irrelevant text used to separate other text parts.
RegExp4	Abbreviation control format vs\\. vb\\. ör\\. Ör\\. [M İ][\\.]*[Ö S]\\. \\d{1,2}\\.
	Used to make the initial abbreviation and ordinal number control.
RegExp5	Roman numbers rule \\b(X{1,3}(IX IV V?I{0,3}) X{0,3}(IX I?V V?I{1,3}))\\b
	Used to indicate roman number format.
RegExp6	SW-Numeric heading format ^\\s*\\d{1,2}[\\. \\) \\-]
	Indicates format of sub-headings that starts with a number Example → “3. Sub heading”
RegExp7	SW-Uppercase heading format ^\\s*[A-H]{1,2}[\\. \\) \\-]
	Indicates format of sub-headings that starts with an uppercase character Example → A- Sub heading
RegExp8	SW-Lowercase heading format ^\\s*[a-h][\\. \\) \\-]
	Indicates format of sub-headings that starts with a lowercase character Example → b) Sub heading

5.3 Software Structure of the Model

Using RegExps that indicate possible sentence boundaries or specific textual formats to catch error-prone conditions is beneficiary, but not sufficient on its own. Considering SBD is combined with different tasks (detection of headings, detection and connection of itemized text parts, assigning a heading for paragraphs) and input files are chosen to be raw text documents, a model to meet the task-specific preprocessing and string control requirements is developed. Figure 5.1 represents the flow diagram of the proposed model.

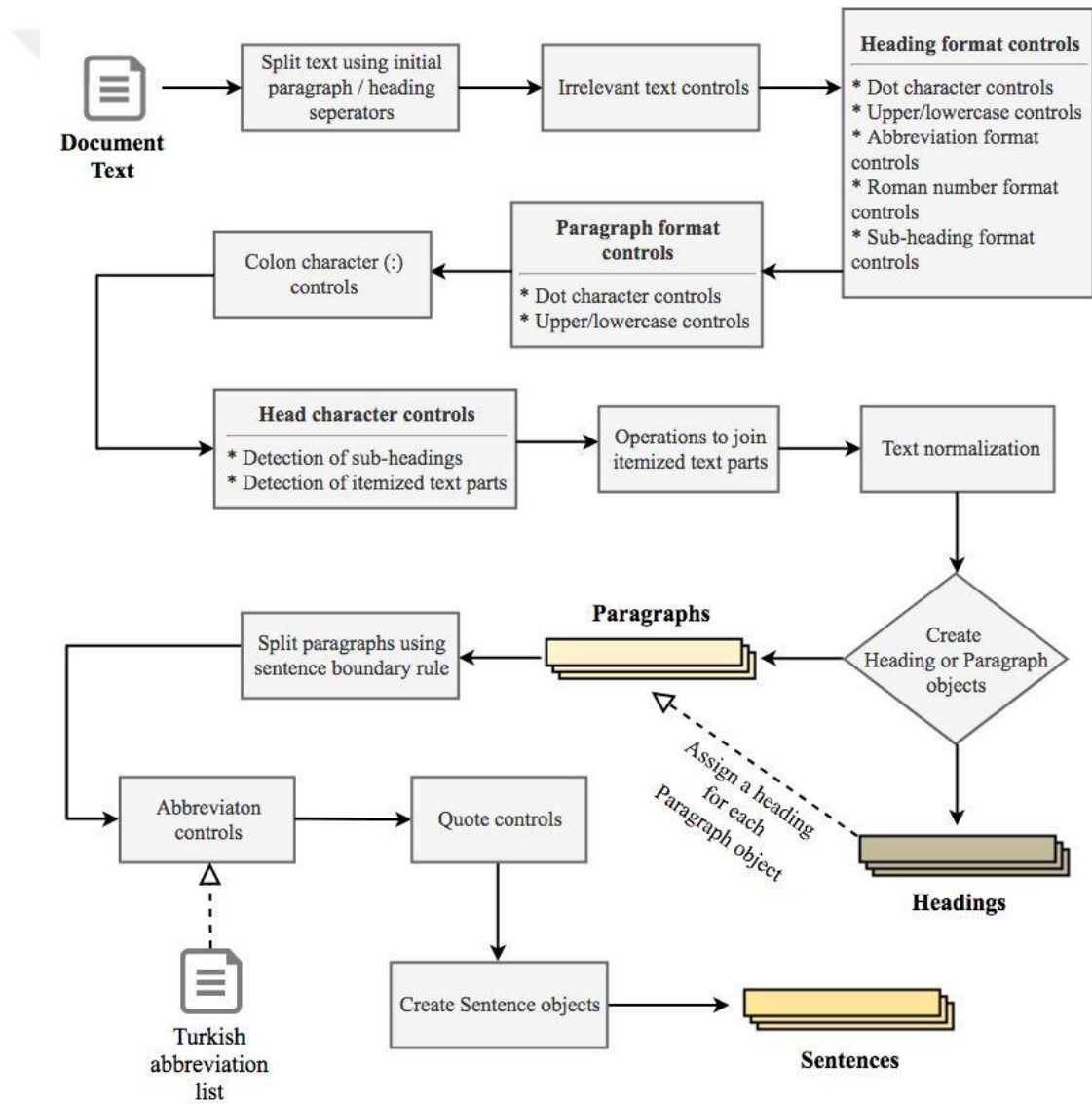


Figure 5.1 Flow diagram of the proposed sentence boundary and heading detection model

5.3.1 Initial Operations

After a text file of lecture notes is loaded to the system and raw text is collected, the filename is assigned as the main heading. Then RegExp2 is used to locate and remove reference formatted text if exists, as they are not needed for the test generation process. Afterwards, a split operation is applied on text content using initial paragraph separators (`\n\n\t`, `\n\n`, `\n\t`, `\r\n\r\n\t`, `\r\n\r\n`, `\r\n\t`), followed by using initial heading separators (`\r\n`, `\n`). `\n` indicates a new line, `\r` indicates a carriage return and `\t` indicates a tab within target text. Split operation returns a list of strings to be worked on, as they are the initial paragraph and heading candidates.

5.3.2 Irrelevant Text Controls

Initial list of strings is checked to eliminate meaningless or non-functional elements. Empty strings, strings consist of whitespace characters only, or strings matched with RegExp3 and detected to be in irrelevant separator format (for example “#####” or “*****”) are extracted from the list in this phase.

5.3.3 Heading Format Controls

Remaining list of strings is checked to find out heading formatted elements. This phase consists of controls for dot characters, upper-lowercase condition of first character and detection of text with abbreviation format, roman number format and sub-heading format. If a string variable doesn't end with a dot and doesn't start with a lowercase character, it must fulfill one of the following conditions to be considered as a heading:

- When inner text parts matched with RegExp4 and RegExp5 + ‘.’ formats are extracted, remaining text should not contain a dot character. Aim here is to prevent erroneous split operations resulted by misleading conditions that contain a dot character like “vs. (etc.)”, “vb. (etc.)”, “III.”, “V.”. Figure 5.2 shows an example system output, where ordinal number (in roman number format) existences within different headings are handled successfully.

- Text should match either a numeric or an uppercase heading format beginning, which are controlled by RegExp6 and RegExp7 rules. Apart from the matched beginning part, text should not contain a dot character.

Raw Text	Headings
I. ve II. Meşrutiyet Dönemlerinin Karşılaştırması	1. ve 2. Meşrutiyet Dönemlerinin Karşılaştırılması (Main)
I. ve II. Meşrutiyet Dönemlerinin Karşılaştırması	I. ve II. Meşrutiyet Dönemlerinin Karşılaştırması (Sub)
I. ve II. Meşrutiyet Döneminin Karşılaştırması	I. ve II. Meşrutiyet Döneminin Karşılaştırması (Sub)
Padişah;	Padişah (Sub)
I. Meşrutiyet	I. Meşrutiyet (Sub)
Yaptıklarından sorumlu değildir.	
Meclisi fesh etme yetkisine sahiptir.	
Mebusan Meclisi başkanını ve Ayan Meclisi	
üyelerini seçme yetkisi vardır.	
Yasama ve yürütme yetkisine sahiptir.	
	Sentences
	(1) Yaptıklarından sorumlu değildir.
	(2) Meclisi fesh etme yetkisine sahiptir.
	(3) Mebusan Meclisi başkanını ve Ayan Meclisi üyelerini seçme yetkisi vardır.
	(4) Yasama ve yürütme yetkisine sahiptir.

Figure 5.2 Detection of headings with ordinal numbers

Additionally, if a string that doesn't end with a dot but starts with a lowercase character matches a lowercase heading format beginning, which is controlled by RegExp8 rule, it is also considered as a heading formatted text. All other conditions violate the heading format in this phase. If a string is not detected to be in heading format, paragraph format controls are applied afterwards.

5.3.4 Paragraph Format Controls

- If a string matches a lowercase heading format beginning and ends with a dot character, it fits the paragraph format.
- If a string starts with a lowercase character but upper condition is not satisfied, it doesn't fit the paragraph format.
- If a string doesn't contain a dot character, it doesn't fit the paragraph format.
- All other conditions are considered to be in a proper paragraph format.

5.3.5 Colon Character Controls

Even though a candidate string fits in paragraph format, it can still contain a sub-heading. Additional operations are needed to detect them, starting with colon character (':') controls. A colon character can either indicate a sub-heading occurrence or just

be a regular text part within a paragraph. Candidate string is split on colon characters for analysis. System approaches on different conditions are explained below:

- If split operation results in multiple strings, text parts that contain a single quotation mark can't be a sub-heading.
- A colon character can be used in a time expression as a delimiter between numeric hour, minute or second values (like "17:39" or "23:59:59"). Divided text parts are joined again when a conforming condition is detected.
- If upper two conditions are not met, heading format controls are applied on each text part. Matched text parts are possible sub-headings. Head character expressions at the beginning of sub-headings (like "A.", "3)", "c-") are removed if any exists.

Figure 5.3 shows an example system output where sub-headings are correctly detected and head character expressions are removed, using colon character controls.

Raw Text	Headings
NEM ve YAĞIŞLAR Atmosfer içerisindeki su buharına nem denir. Nem higrometre adı verilen aletle ölçülür. Havanın nemi gram (gr) olarak ifade edilmektedir. 1. Mutlak Nem: 1m3 hava içerisinde bulunan suyunun gr olarak ağırlığına mutlak nem denir. Mutlak nem, sıcaklık ve buharlaşmanın fazla olduğu Ekvatorial bölgelerde çok, soğuk kutup bölgeleri ile yüksek dağlarda azdır. 2. Maksimum Nem: 1m3 havanın belli sıcaklıkta taşıyabileceği en fazla nem miktarına maksimum nem denir. Maksimum nem sıcaklığa bağlı olarak değişir. Sıcaklık arttıkça hava genişleyeceğinden taşıyabileceği nem miktarı artar. Sıcaklık azaldıkça hava daralır ve böylece taşıyabileceği nem miktarı azalır. Sıcaklıkla maksimum nem doğru orantılıdır. 3. Bağıl Nem (Nisbi nem): Mutlak nemin maksimum neme oranı havanın neme doyma oranını verir. Bu orana bağıl nem denir.	Nem, Yoğunlaşma ve Bulutlar (Main) NEM ve YAĞIŞLAR (Sub) Mutlak Nem (Sub) Maksimum Nem (Sub) Bağıl Nem (Nisbi nem) (Sub) YOĞUNLAŞMA (Sub) Yüksek bulutlar (Sirüs'ler) (Sub) Orta yükseklikteki bulutlar (Kümüls'ler) (Sub)
	Sentences (1) 1m3 havanın belli sıcaklıkta taşıyabileceği en fazla nem miktarına maksimum nem denir. (2) Maksimum nem sıcaklığa bağlı olarak değişir. (3) Sıcaklık arttıkça hava genişleyeceğinden taşıyabileceği nem miktarı artar. (4) Sıcaklık azaldıkça hava daralır ve böylece taşıyabileceği nem miktarı azalır. (5) Sıcaklıkla maksimum nem doğru orantılıdır.

Figure 5.3 Detection of sub-headings at the beginning of paragraphs

5.3.6 Head Character Controls

Using itemization to present information via a list of contextually connected text parts is a common approach in lecture notes. However, itemized text parts can mistakenly be considered as sub-headings if they don't contain a dot character. This requires a disambiguation phase for all string objects in the list that are previously detected as a sub-heading candidate.

- If consecutive strings obey the same sub-heading format rule (RegExp6, RegExp7 or RegExp8), they are considered as itemized text parts.
- If consecutive strings with sub-heading format start with the same head character symbol (like “#, *, ♦, ►, ○, ●, →, □” etc.), they are considered as itemized text parts. A total of 27 head character symbols are defined in the system.
- If a string with sub-heading format ends with a comma character, it is a probable itemized text part and needs to be joined with the following string in the list.

Figure 5.4 shows an example system output, where sub-heading formatted itemized text parts are correctly handled and joined together, as they share the same RegExp6 format. Join operations are detailed on the next section.

Raw Text	Headings
NÜFUS SAYIMLARI SONUCU ELDE EDİLEN VERİLER	Nüfus Sayımları Sonucu Elde Edilen Veriler (Main)
Nüfus sayımları sonucunda elde edilen veriler;	NÜFUS SAYIMLARI SONUCU ELDE EDİLEN VERİLER (Sub)
1. Nüfus miktarı	Nüfus sayımları sonucunda elde edilen veriler (Sub)
2. Nüfus artış hızı ve dağılışı	
3. Nüfusun yaş ve cinsiyet durumu	
4. Nüfusun kır-kent yerleşimlerine göre dağılımı	
5. Aktif nüfusun sektörel dağılımı	
6. Nüfusun eğitim durumu	
7. İşsizlik oranı	
8. Nüfus hareketleri	
9. Askerlik çağındaki nüfus	
10. Seçmen sayısı	
	Sentences
	(1) Nüfus miktarı, nüfus artış hızı ve dağılışı, nüfusun yaş ve cinsiyet durumu, nüfusun kır-kent yerleşimlerine göre dağılımı, aktif nüfusun sektörel dağılımı, nüfusun eğitim durumu, işsizlik oranı, nüfus hareketleri, askerlik çağındaki nüfus, seçmen sayısı.

Figure 5.4 Detection of sub-heading formatted itemized text parts

5.3.7 Operations to Join Itemized Text Parts

If disambiguation phase resulted with detection of itemized text parts, a join operation using comma characters is performed. To transform the first character of an itemized string to lowercase or not is one decision to make on this phase. If first word of the string contains an apostrophe character, or second word starts with an upper character, this implicates first word being a proper name, so its first character is not transformed and left as an uppercase character. In all other conditions, first character is converted into lowercase. This operation is critical for named entity recognition (NER) module success.

Where to start and stop the join operation is the other decision to make. Initially, all of the detected itemized text parts are joined using a comma character and a combined string is obtained, but preceding and following string objects are also checked to complete join operation.

- If the preceding string is not detected as a sub-heading, combined string is appended to this string using a whitespace character as delimiter.
- If the following string starts with a head character symbol and ends with a dot, it is appended to the combined string after the symbol is removed, using a whitespace character as delimiter.
- If the following string begins with the same sub-heading format used in itemized text parts, and ends with a dot, it is appended to the combined string after head characters are removed, using a whitespace character as delimiter.
- If combined string consists of itemized text parts each ends with a comma character:
 - If following string fits the paragraphs format, it is appended to the combined string using a whitespace character as delimiter.
 - If following string doesn't fit the paragraph format, last character of combined string is transformed into a dot.

Figure 5.5 shows an example system output for itemized text parts join operation. As preceding string is not a heading, combined string is appended to this text. Besides, first characters of each itemized text are properly transformed into lowercase, as none of the first words expresses a proper noun.

<p>geçirip yönettiği ülkeye veya topraklara koloni denir.</p> <p>Kolonilerin kurulmasında:</p> <ul style="list-style-type: none"> • Hammadde ihtiyaçlarının karşılanması • Üretim fazlası mallar için pazar bulunması • Askeri gücün arttırılmak istenmesi • Diğer devletlere askeri, siyasal ve ekonomik alanlarda üstünlük sağlama düşüncesi etkili olmuştur. 	<p>Sentences</p> <p>sınırları dışında ele geçirip yönettiği ülkeye veya topraklara koloni denir.</p> <p>(69) Kolonilerin kurulmasında; hammadde ihtiyaçlarının karşılanması, üretim fazlası mallar için pazar bulunması, askeri gücün arttırılmak istenmesi, diğer devletlere askeri, siyasal ve ekonomik alanlarda üstünlük sağlama düşüncesi etkili olmuştur.</p> <p>(70) Anadolu'ya yazıyı Mezopotamya medeniyetlerinden Asurlular getirmiştir.</p>
---	--

Figure 5.5 Example itemized text parts join operation output

5.3.8 Text Normalization

Text normalization is the last phase before creating Paragraph and Heading objects from candidate strings. Main reason for not doing normalization up to this point is to benefit from suggestive textual cues as much as possible. Included adjustments in text normalization are stated below:

- Hyphens on new line beginnings are removed.
- New lines and tabs are removed.
- Control characters are removed.
- Irrelevant characters like symbols are removed.
- Triple dot characters are translated into a single dot character, if any exists.
- Multiple adjacent whitespaces are replaced with a single one.

5.3.9 Generation of Heading and Paragraph Objects

After the normalization of a candidate string is completed, final classification decision is given on adjusted text and either a Heading or Paragraph class object is created. Information about these two classes are shown on Table 5.3. This operation is repeated for every candidate string. Thus, two separate lists for paragraphs and headings are obtained.

Besides, every Paragraph object is correlated with a Heading object. If the input document text doesn't contain any sub-headings, then the main heading (input filename) is assigned to every Paragraph object. This task is the main reason for not separating paragraph and heading detection jobs, as running them together preserves the order of text parts and makes the heading assignment operation easier, as nearest preceding heading is searched. Figure 5.6 shows the logic behind this operation.

Table 5.3 Heading and Paragraph class fields with their types

Class Name	Class Fields
Heading	ID (int), Text (string), IsMain (bool), IsSub (bool)
Paragraph	ID (int), Text (string), OwnerTitle (Heading)

<p>saat çoğunluk, TBMM'nin işleyişi, amacı ve süresi konusunda belirlemelerde bulunulmuştur. Bunlarla yetinmeyen BMM temel haklar komisyonu oluşturmuş ve bu komisyonun hazırladığı "Büyük Millet Meclisi'nin Kuruluş ve Niteliği" ile ilgili yasa taslağını görüşmeye başlamıştır. Bu taslağın 1. maddesi "Büyük Millet Meclisi yasama ve yürütme güçlerini kendinde toplar ve devleti bağımsız olarak yönetir." hükmü tutucu milletvekilleri tepkiyle karşılanmıştır.</p>	<p>çoğunluk, TBMM'nin işleyişi, amacı ve süresi konusunda belirlemelerde bulunulmuştur. (10) Bunlarla yetinmeyen BMM temel haklar komisyonu oluşturmuş ve bu komisyonun hazırladığı "Büyük Millet Meclisi'nin Kuruluş ve Niteliği" ile ilgili yasa taslağını görüşmeye başlamıştır. (11) Bu taslağın 1. maddesi "Büyük Millet Meclisi yasama ve yürütme güçlerini kendinde toplar ve devleti bağımsız olarak yönetir." hükmü tutucu milletvekilleri tepkiyle karşılanmıştır. (12) Hükümet, 18 Eylül 1920'de de meclise bir anayasa tasarı ve bu tasarıya gerekçe özellikleri taşıyan halkçılık programı getirmiştir.</p>
---	---

Figure 5.8 Output when a not-sentence boundary case is detected between a dot and an apostrophe

5.3.10.2 Abbreviation Controls

Although an initial abbreviation control is done on earlier stages, abbreviations with [UPPER | lower] [lower]+ [\.] format are still prone to error as they fit the sentence boundary rule. To overcome this, publicly available abbreviation list of Turkish Linguistic Association (Türk Dil Kurumu, TDK) is scanned for a subset that matches with the problematic format. Finally, a Turkish abbreviation list with a total of 204 elements is generated for the system usage. Complete list is given in Appendix-3.

Candidate sentence strings are checked for abbreviations in the list and if any match is found, abbreviation text is joined with the following string in the list. Besides, some abbreviations consist of multiple words (like "Dz. Kuv. K." (Commander of Naval Forces) require to be regrouped before a join operation.

5.3.10.3 Quote Controls

Last operation before generation of Sentence objects is quotes controls. System approach is to avoid dividing inner-sentence quotes even if they state one or more sentence boundaries. Main goal of this approach is to preserve content integrity and provide better question quality for further phases.

Number of quotation marks is the decision metric used. A candidate string is not considered as a sentence if it contains odd number of quotation marks and joined with following strings until number of quotation marks in the combined string becomes an even number. This operation is applied just for inner-sentence quotes and quotes of a whole paragraph are divided based on the initial sentence boundary detections.

Figure 5.9 and Figure 5.10 show how system output changes when inner-sentence quote controls are disabled or enabled.

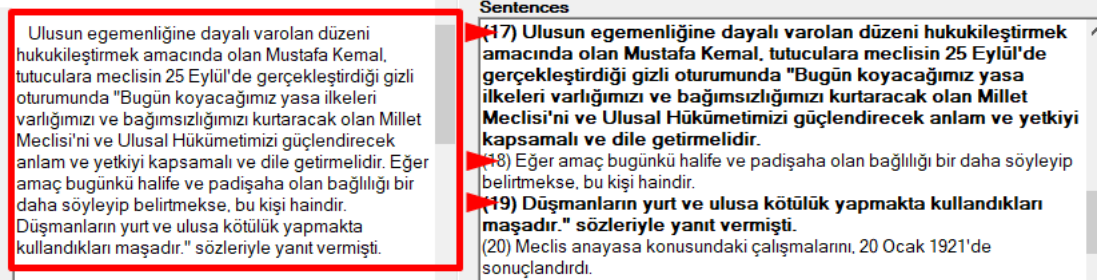


Figure 5.9 Example output when inner-sentence quote controls are disabled

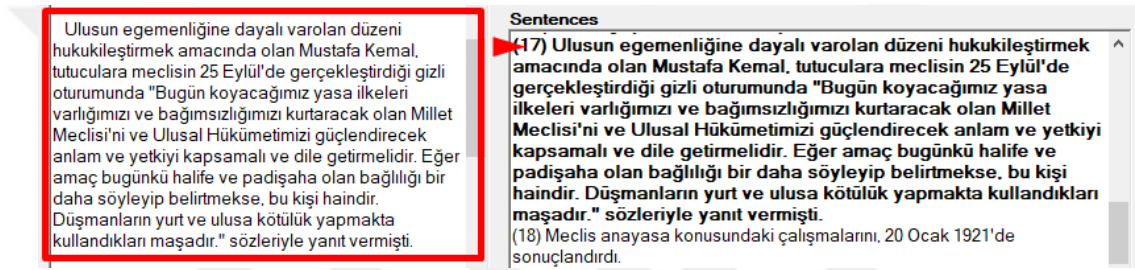


Figure 5.10 Example output when inner-sentence quote controls are enabled

5.3.10.4 Generation of Sentence Objects

Using the final forms of remaining candidate strings, Sentence objects are created. Sentence class consists of Text (string), OwnerTitle (Heading) and EndsWithDot (bool) fields. Owner title information of a recently created Sentence object is taken from the outer Paragraph object's field with the same name. Eventually, a list of sentences is obtained from an input text document.

5.4 Experimentation Phase

5.4.1 Used Dataset

Success of the system is tested via experiments on actual lecture notes. 30 history and 30 geography documents are selected out of the primary lecture notes dataset which is introduced in section 4.1.

5.4.2 Sentence Boundary and Heading Detection

Precision and recall metrics are used to evaluate both sentence boundary detection and heading detection success. Experiments for geography domain and history domain are separated to allow comparisons and conclusive results are calculated by combining these two experiment sets. Precision values are calculated by dividing number of correct guesses to number of all detections, recall values are calculated by dividing number of correct guesses to number of actual occurrences. Evaluation metrics used on experiments are formulated below on equations 5.1, 5.2, 5.3 and 5.4. Combined experimental results are shown on Table 5.4 and Table 5.5.

$$\text{Precision Sentences (\%)} = \frac{100 (\# \text{ of Correct Sentences})}{\# \text{ of Detected Sentences}} \quad (5.1)$$

$$\text{Precision Headings (\%)} = \frac{100 (\# \text{ of Correct Headings})}{\# \text{ of Detected Headings}} \quad (5.2)$$

$$\text{Recall Sentences (\%)} = \frac{100 (\# \text{ of Correct Sentences})}{\# \text{ of Actual Sentences}} \quad (5.3)$$

$$\text{Recall Headings (\%)} = \frac{100 (\# \text{ of Correct Headings})}{\# \text{ of Actual Headings}} \quad (5.4)$$

Table 5.4 Suggestive numerical values derived from SBD and heading detection experiments

DOMAIN	# of Actual Sentences	# of Detected Sentences	# of Correct Sentences	# of Actual Headings	# of Detected Headings	# of Correct Headings
History Documents (30)	1241	1233	1208	175	173	170
Geography Documents (30)	1093	1089	1086	292	290	290
TOTAL	2334	2322	2294	467	463	460
AVG	38.90	38.70	38.23	7.78	7.72	7.67

Table 5.5 Precision and recall values derived from SBD and heading detection experiments

DOMAIN	Precision Sentences (%)	Recall Sentences (%)	Precision Headings (%)	Recall Headings (%)
History Documents (30)	97.97	97.34	98.27	97.14
Geography Documents (30)	99.72	99.36	100.00	99.32
TOTAL	98.79	98.29	99.35	98.50

As Table 5.4 shows, average number of sentences in the combined dataset documents is stated as 38.90 and 38.23 of them are correctly detected. When domain

specific experiments are observed, average number of sentences is stated as 41.37 for history course and 36.43 for geography course documents, while 40.27 and 36.20 of them are correctly detected respectively.

Average number of headings in the combined dataset documents is stated as 7.78 while 7.67 of them are correctly detected. Domain specific results show that average number of headings is stated as 5.83 for history course and 9.73 for geography course documents, while 5.67 and 9.67 of them are correctly detected respectively.

Based on these values, it is possible to say that most history course text documents tend to contain lower numbers of headings and higher numbers of sentences compared to geography course documents.

Table 5.5 shows the individual and combined experiment results based on the selected performance metrics. Experiments on history course text documents resulted in 97.97% precision for sentences, 97.34% recall for sentences, 98.27% precision for headings and 97.14% recall for headings. Experiments on geography course text documents resulted in 99.72% precision for sentences, 99.36% recall for sentences, 100% precision for headings and 99.32% recall for headings.

Combined results are stated as **98.79%** precision for sentences, **98.29%** recall for sentences, **99.35%** precision for headings and **98.50%** recall for headings.

Results show that success rate for geography domain is slightly better than history domain in all metrics. This difference is understandable, as the amount of textual context in a history document is observed to be more than a geography document and this increases the variety in text and uncommon use cases.

Apart from that, precision values are determined to be slightly higher than recall values on same experiment sets. But in general, results are satisfactory, especially if the approach to combine different tasks is considered.

5.4.3 Observation of Itemized Text Part Cases

Detection and connection of itemized text parts is one of the sub-tasks the model deals with. Correctly making the upper-lowercase adjustments for the first characters of text parts on join operation is also an important job within this task, especially for NER module success. Table 5.6 shows the experiment results.

Table 5.6 Experiment results of itemized text parts detection and join operations

DOMAIN	# of Cases to Join Itemized Text Parts	# of Correctly Detected Cases	# of Missed Cases	# of Detected Items	# of Items Correctly Converted to Lower or Upper Case	# of Items Incorrectly Converted to Lower or Upper Case
History Documents TOTAL (30)	14	14	0	81	71	10
Geography Documents TOTAL (30)	13	13	0	77	77	0
TOTAL	27	27	0	158	148	10

Results show that all of the 27 cases where a join operation is needed on itemized text parts are correctly detected. First character upper-lowercase adjustments are correctly handled on 148 text parts out of the total 158 detected. All 10 incorrectly converted cases are resulted from single-word proper nouns in nominative case, as neither a following apostrophe or following word with uppercase clue exists.

CHAPTER SIX

VERB POLARITY DETECTION AND CONVERSION

6.1 Overview

Two consecutive tasks will be detailed on this chapter: Verb polarity detection (which will be mentioned as “classification” after this point) phase and conversion phase. Classification phase is where the input sentence is analyzed and classified as affirmative (positive) or negative. Conversion phase is where the classified sentence is translated into the corresponding opposite form. Both tasks deal with the verb of the given sentence. As it is a morphological approach, semantics is not the concern. For example, the Turkish word “*endişelenmek*” (to be worried) is semantically negative, but the sentence “*Senin için endişelendim.*” (I am worried about you) is morphologically positive and classified as an affirmative sentence.

Stem type identification is an essential sub-task within this model, which aims to correctly label the predicate (main verb) of a sentence as verb-typed or noun-typed, after a morphological analysis phase.

General flow of the algorithm is like below:

- I. Take the verb of the input sentence.
- II. Do the classification operations and decide whether the sentence is affirmative or negative.
- III. If AFFIRMATIVE: Convert the sentence into the corresponding negative form.
- IV. IF NEGATIVE: Convert the sentence into the corresponding affirmative form.

Algorithm is implemented in a way to deal with canonical sentences (in which the predicate is located at the end of the sentence). So, the input lecture notes are expected to contain meaningful and canonical sentences for proper results. Also, using UTF-8 encoding in text files is recommended to handle special Turkish characters correctly.

6.2 Classification Phase

Turkish is a complex language with its grammar and phonetic rules. As it is an agglutinative language, a bare infinite (base) form of a verb can gain additional meanings with the addition of several suffixes, like subject, tense etc. Implemented algorithm deals with verb polarity, which is one of these derivable meanings.

As mentioned on Chapter One, Turkish words might have relatively long words with 9-10 affixes. Nevertheless, circumstances are much more reasonable in general. Studies on Turkish language specifies the average number of morphemes per word as 3 (including the root), while high-frequency words usually have a single morpheme. Average number of morphological interpretations per word in written language is specified as 2 on same studies, while 65% of words have a single morphological interpretation (Ofllazer, 2014). Additionally, there exists phonetic rules for certain circumstances with few irregularities and grammatical rules for suffixes. Due to these reasons, rule-based approaches are formed the basis of this task.

6.2.1 Finite State Machine Structure

To handle the classification task, a Finite State Machine (FSM) structure is implemented. Main idea on that is to teach the system all possible verb + suffix combinations and expect it to give correct output. The FSM in this project shows the rules to combine different types of suffixes and a verb/noun stem to form a morphologically correct verb. The reasons why FSM is selected as the model for this process are listed below:

- I. Compatibility with the agglutinative aspect of Turkish.
- II. Running once at the execution of the program is enough for further usage.
- III. Simplicity in modification and add-delete data operations.
- IV. The repetitive parts of different rules are joined where possible, which minimizes the rules to learn.
- V. Provides a convenient environment to find the verb's stem.

Determined rules on FSM are interpreted from the verb's right to left to reach the stem of the word. So, in the classification phase, the stem of the verb, which will be used in the conversion phase afterwards, is also found besides the verb polarity. Main FSM elements and how they are adapted for the model usage is detailed on Table 6.1.

Table 6.1 Main FSM elements and their purposes

Name	Purpose
State	<p>Different conditions are represented with different states in the FSM. Each state stores information about the possible past actions that lead to that state and possible future actions allowed on the model. The machine can only be in one state at a time. Start state is the initial state where the machine is started. Goal state is the state that indicates that the input string, as processed so far, is in a form that that the machine language accepts. Final state is a goal state with a dead end, which means FSM can't move any further from these states. In this model, start state is the initial form of a given word and final (or goal) states are the conditions where the verb fits the provided morphological rules, therefore a possible word stem is reached.</p> <ul style="list-style-type: none"> - 50 states are used in this model (18 of them are goal states and 6 of them are final states).
Input	<p>A triggering event or a condition, which leads the machine to move from a state to another state. In this project, inputs are suffixes that can be attached to a verb or noun stem to form a predicate.</p> <ul style="list-style-type: none"> - 35 suffixes are defined and used in this model.
Transition	<p>A connection between two state variables via an input variable. So, a transition consists of a start state, input value and a finish state. Transitions are main elements to generate rules to control whether a given word fits the predicate format in terms of Turkish morphology or not.</p> <ul style="list-style-type: none"> - 263 transitions are used in this model.

Figure 6.1 shows an example FSM execution on the word “*okutmalıldrlar*” (They should educate (them)), in which system is reached one of the final states. A is the start state, F3 is the final state, C and T are other states involved.

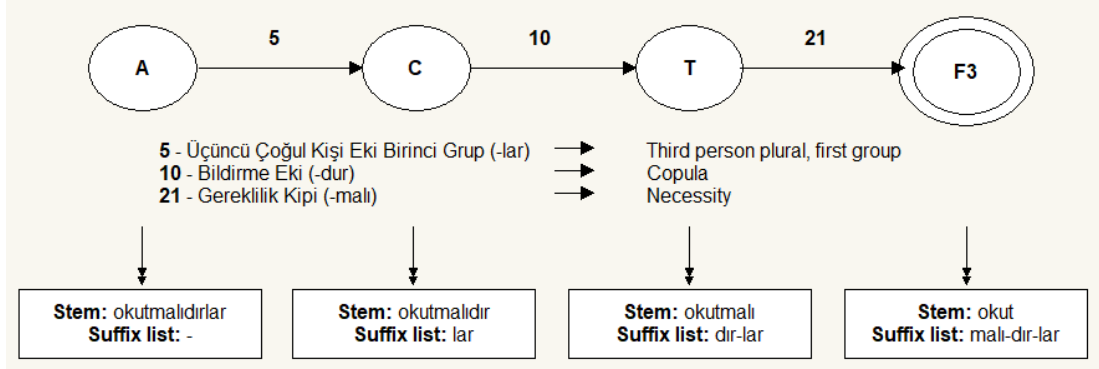


Figure 6.1 Sample FSM execution on a word with a proper predicate format

Example word given in Figure 6.1 is examined by FSM with following steps:

- Suffix “-lar” moves the machine from A to C state.
Stem: *okutmalıdır*
- Suffix “-dur (-dır)” moves the machine from C to T state.
Stem: *okutmalı*
- Suffix “-malı” moves the machine from T to F3 state, the process is stopped.
Stem: *okut*

Since F3 state is a final state, the word “*okutmalıdır*” is parsed in a form “*okut-malı-dır-lar*” and the possible stem is found as “*okut*” (To educate/teach sb.) in this example.

Main purpose is to find the stem of the verb, not root. Because derivational suffixes are not useful for conversion phase, but inflectional suffixes are. For example “*okut*” is the desired result on the upper example, not “*oku*” (to read or to study). “-t” is a derivational suffix here that changes the meaning of the word.

The vowels of the same suffix may vary in different words, so actually an input value is a collection of suffix values. For example, the input value 5 (“Üçüncü Çoğul Kişi Eki Birinci Grup” – Third Person Plural, First Group) can indicate “-lar”, “-ler”. In the same manner, input value 10 (“Bildirme Eki” – Copula) can indicate “-dir”, “-dır”, “-dur”, “-dür”. The diversify rules depend on the first vowel of the suffix, which is showed on Table 6.2.

Table 6.2 Rules to diversify suffixes regarding the first vowel of the suffix

First Vowel (To Variate)	Alternative Vowels
i	i, í, u, ü
a	a, e

6.2.2 Base Class Definitions

6.2.2.1 State

State class objects are used to represent current condition during a verb parse operation. Changes on states depend on the found suffix at the end of the word. Each change means a suffix match is found and the word will be parsed. Consequently, every parse operation generates a shorter stem value. Table 6.3 shows the State class fields along with their types and purposes.

Table 6.3 Description of State class fields

Field Name	Type	Description
Name	string	Used to assign a unique name for each State variable. Most of them are a single character. Some exceptions are final states (like F1, F2), states of infinitive suffix conditions (like MA, MB) and states of ability suffix conditions (like YA, YB).
IsGoal	bool	Indicates that whether the state is goal state or not. Note that every final state is a goal state, but every goal state may not be a final state. Final states are named as F1, F2, F3, F4, where the digit at the end implies the level of state. Final states are absolute dead ends for the FSM, while suffix parse operations can continue from other goal states.
Level	integer	Indicates the number of steps to reach to that particular state from start state.

6.2.2.2 Suffix

Suffix class objects are used to hold information of the (mostly inflectional) suffixes that can generate a predicate when appended to a word in bare infinitive form in certain conditions. Suffix objects are the input variables for FSM. Table 6.4 shows the Suffix class fields along with their types and purposes.

Table 6.4 Description of Suffix class fields

Field Name	Type	Description
Definition	string	An explanatory name for the suffix. Examples: <ul style="list-style-type: none">- “<i>SimdikiZamanEki</i>” (Present continuous tense suffix)- “<i>GelecekZamanEki</i>” (Simple future tense suffix)
PositiveValues	List<string>	Holds the possible affirmative (positive) values for the suffix. <ul style="list-style-type: none">- For <i>SimdikiZamanEki</i>: { “(i)yor”, “(u)yor”, “(ı)yor”, “(ü)yor” }
NegativeValues	List<string>	Holds the possible negative values for the suffix, if exists. <ul style="list-style-type: none">- For <i>SimdikiZamanEki</i>: { “miyor”, “muyor”, “mıyor”, “müyor” }
Format	string	Holds a format string to specify the changeable and helper characters of the suffix, for the situations where phonetic controls are necessary. Changeable characters are shown with a dot, helper characters are shown between parenthesis. <ul style="list-style-type: none">- For <i>SimdikiZamanEki</i>: “.yor”- For <i>GelecekZamanEki</i>: “(y).c.k”

6.2.2.3 Transition

Transition class objects are used to connect State variables with an input Suffix variable. Transitions are used to store predicate format rules and essential to find out possible stems of the input word. Table 6.5 shows the Transition class fields along with their types and purposes.

Table 6.5 Description of Transition class fields

Field Name	Type	Description
StartState	State	The initial state for the transition.
InputSuffix	Suffix	The trigger suffix value which changes the initial state.
FinishState	State	When a triggering suffix value is perceived, current state value is changed from StartState to FinishState and this this transition is completed.

6.2.2.4 SuffixInWord

SuffixInWord class can be considered as an extended version of Suffix class. Class objects are used after a transition is succeeded and aims to provide some disambiguation for the verb polarity classification operation. Table 6.6 shows the SuffixInWord class fields along with their types and purposes.

Table 6.6 Description of SuffixInWord class fields

Field Name	Type	Description
Suffix	Suffix	The Suffix object found in the given word and leads to a transition execution.
Index	Integer	The index value of the found suffix among the all possible values of the suffix.
IsPositive	Bool	Is set true if the found suffix value is one of the affirmative (positive) values.
IsNegative	Bool	Is set true if the found suffix value is one of the negative values.

6.2.2.5 Path

A SuffixInWord object holds information after a successful transition execution. However, it doesn't provide enough information for a complete FSM process. Path class is defined for this purpose, as it holds the complete information about a verb parse and classification operation. Every allocated Path variable offers a possible solution for the verb parse and polarity classification task. Table 6.7 shows the SuffixInWord class fields along with their types and purposes.

Table 6.7 Description of Path class fields

Field Name	Type	Description
StatesString	string	Holds the names of all states where the FSM passed through, from the start state up to current state.
CurrentState	State	Holds the current state variable of the path.
WordRoot	string	The part of the word which is separated from the suffix part. (Stem or Root)
WordSuffixPart	string	Holds the suffix part of the word with using a separator '-' between suffixes.
WordSuffixList	List<SuffixInWord>	Holds the detailed information about each detected suffix in the word.
TranslateIndex	integer	Shows the index of the suffix in WordSuffixList, which should be translated in conversion phase.
IsRootNoun	bool	Is set true if the word class of the found stem (or root) is noun.
IsRootVerb	bool	Is set true if the word class of the found stem (or root) is verb.
IsRootBothNameAndVerb	bool	Is set true if the word class of the found stem (or root) may be a noun or verb.
IsPositive	bool	Is set true if the Path object classifies the given predicate as positive.
IsNegative	bool	Is set true if the Path object classifies the given predicate as negative.

6.2.2.6 *FSM*

FSM is a static class where the FSM initialization operations are executed and suffix list with 35 elements, state list with 50 elements and transition list with 263 elements are prepared for usage. Two lists to contain exceptional verb roots, which will be detailed later, are also stored within this class.

6.2.3 *Parse Operations*

After an input text file is loaded to the system and FSM initialization is completed, parse operations are started to detect verb polarity and the most suitable stems of predicates. Each parse operation works on a sentence which are obtained from sentence boundary module. The step-by-step process for the parse operations are given below:

- 1) Mutual variables which are used on each iteration are reset before a new sentence parse operation.
- 2) The predicate of the sentence is found on the assumption that user loaded a text file with meaningful and canonical sentences.
- 3) Start state is set as current state.
- 4) Initial predicate value is assigned as the stem candidate.
- 5) FSM is activated by scanning through transition list considering current state.
- 6) If a transition's start state matches with the current state, stem candidate is sent for the suffix control.
- 7) If the stem candidate ends with the transition's input suffix, current state is set as the transition's finish state and process continues with Step 9.
- 8) If the predicate doesn't end with the transition's input suffix, process returns to Step 5.
- 9) The predicate is separated from the found suffix, which is added to the current word's suffix list. A new stem candidate is derived by removal of found suffix.
- 10) If the new current state is a goal state (but not a final state), process stops by Step 13 for verb polarity detection and Path object creation, then returns to Step 5 by sending the stem candidate to FSM again.

- 11) If the new current state is a final state, process continues with Step 13 for verb polarity detection and Path object creation, then it's completed for this condition.
- 12) If the new current state is not in a goal state, process returns to Step 5 by sending the stem candidate to FSM again.
- 13) Verb polarity classification is done regarding to IsPositive and IsNegative fields of the SuffixInWord object created for the last detected suffix. Using collected information, a new Path object is created and added to the result path list.
- 14) After all operations are completed, a number of paths are obtained.

All of the resulted paths are possible solutions, as they are in a format that obeys the rules provided from transitions, and the reached state is a goal state. Parse operations are completed, but the number of paths should be reduced before the conversion phase if possible.

6.2.4 Path Elimination Operations

6.2.4.1 Elimination of Irrelevant Results

Actually, this step is used to prevent creation of incorrect paths, not to eliminate already created paths. To show effects of all consecutive path reduction operations, it is explained under this section. After a candidate stem is found with a transition execution on FSM, if the new current state is not a goal state, process continues to scan using new candidate stem, but a Path object is not created at this point. For example, for the predicate “*kaçacağım*” (I will flee/escape), 4 irrelevant results are eliminated out of the 6 initial results, as they are not in a goal state.

6.2.4.2 Elimination of Meaningless Stems

For the next two steps, system benefits from the database structure built in the study of Aktaş (2010). As detailed in Chapter 3, database model is built using a complete lexicon with the list of words in Turkish dictionary, provided by TDK on that study.

Govde table stores a stem list for Turkish which is a serviceable resource on this elimination step. Remaining paths, which don't exist in *Govde* table are detected and eliminated. “*kaçacağ-ım*”, which is one of the two remaining paths from the upper example is eliminated on this step. Figure 6.2 shows the system output for the given example and depicts how FSM execution for suffix parse operations and following path elimination steps work on verb polarity classification and stem detection tasks.

Input Sentence / Word		
kaçacağım		
<input type="button" value="Classify"/> <input type="button" value="Convert"/>		
Classification		
First Results From FSM	Results in Goal States	Results with Meaningful Stems
kaçacağ-ım (State: D) kaçacağ-ım (State: N) kaçacağ-ım (State: Q) kaç-acağ-ım (State: F2) kaçacağ-ım (State: YJ) kaç-acağ-ım (State: YH)	kaçacağ-ım (State: Q) kaç-acağ-ım (State: F2)	kaç-acağ-ım (Affirmative)

Figure 6.2 Example system output for verb polarity classification and stem detection tasks

6.2.4.3 Elimination by Root Type

Another path elimination method in classification phase is done by root type restraints. *Govde* table also holds Boolean attributes to indicate possible POS tag info with related attributes like isNoun, isAdj, isAdv. Some goal states in FSM are meaningful for only noun or verb root types, so *Govde* table is used to find out the candidate types of remaining stems (or roots) in search of extractable paths. For example, Turkish root “*düş*” can indicate either a noun (“dream” in English) or verb (“to fall” in English). But it can't be a verb in the predicate “*düş-tür*” (It is a dream), while it can't be a noun in “*düş-müş-tür*” (He/She must have fallen).

6.2.4.4 Elimination by Supervised Learning

In some conditions, found stem (or root) might be either a verb or noun and both conditions are meaningful and in a goal state. A probabilistic supervised learning

approach is used here to overcome the ambiguity in those cases. A basic database table is created to store numbers of verb and noun occurrences of an ambiguous word. While training the model with real text-based lecture notes, every time when this type of an ambiguity occurs, noun/verb decision for the problematic root is given, then its related index in the database was increased. Results show that noun occurrences of ambiguous word roots are very few, even neglectable for this project's domain. For example, root “*düş*” within the predicate “*düştü*” (which means “He/She/It has fallen” with verb root, or “It was a dream” with noun root) is prone to be classified as verb, as it is used as a verb-typed root in all 24 occurrences of this predicate within the dataset. So, system is designed to eliminate paths that determined the root to be a noun when this kind of ambiguations are encountered.

6.3 Conversion Phase

Up to now, the program took a text file from the user, separated it to sentences on sentence boundary detection (SBD) module, used FSM structure as a morphological analyzer to parse the sentences and get candidate paths, eliminated the unsuitable paths and classified remaining paths in terms of verb polarity (as affirmative or negative). Conversion phase continues with the remaining paths and aims to find out the optimal result. The information needed for a complete conversion of a path (which is a probable solution for a particular sentence) is the found stem of the predicate, classification result for the predicate, suffix list of the path, index of the suffix to be translated in the suffix list, type and format of that suffix, opposite values of that suffix and the phonetic rules to be considered.

6.3.1 Controls for Phonetic Rules

The main complexity of the conversion phase is the obligation of dealing with the Turkish phonetic rules. Sometimes it may be enough to simply change the suffix value with its opposite form to fulfill the conversion, but it's a very rare condition. Effects of the phonetic rules can be generalized in two main categories: The stem of a predicate can be changed with the addition of a suffix, or the suffix needs to be formed according to stem of the predicate and previously appended suffixes to the stem.

Table 6.8 briefly explains the Turkish phonetic rules considered within the model, while Table 6.9 describes auxiliary terms to control phonetic rules and Table 6.10 introduces the phonetic rule control methods defined in the program.

Table 6.8 Phonetic rules considered in the program

Name	Description	Example
Consonant assimilation (<i>Ünsüz benzeşmesi</i>)	If the stem ends with a strong consonant (<i>sert ünsüz</i>), first letter of the suffix is changed to ‘t’ if it is initially ‘d’.	<i>kariş + (y)dı</i> <i>kariş + tı</i> <i>karişti</i> (It is mixed/joined)
Consonant lenition (<i>Ünsüz yumuşaması</i>)	If the stem ends with the strong consonant ‘k’, it is changed to ‘ğ’, when stem is a noun type and the suffix starts with a vowel. Also, the ending strong consonant ‘t’ is changed to ‘d’ on some exceptional verb roots (like “ <i>git</i> ” (go), “ <i>farket</i> ” (realize)) in certain conditions. These verb roots are stored under a string list defined on FSM class, also given in Appendix-4.	<i>kaçak + (y)ım</i> <i>kaçığ + ım</i> <i>kaçığım</i> (I am fugitive) <i>farket + (e)r</i> <i>farked + er</i> <i>farkeder</i> (He/She realizes)
Helper letter addition in front of the suffix	Controls to add the helper letter ‘y’ at the beginning of the suffix, when the stem ends with a vowel.	<i>uyu + (y)acak</i> <i>uyuyacak</i> (He/She will sleep)
Removal of the first letter of suffix	Controls to remove the first letter of suffix, when the root ends with a vowel.	<i>uyu + (i)yor</i> <i>uyu + yor</i> <i>uyuyor</i> (He/She is sleeping)
Vowel reduction (<i>Ünlü daralması</i>) on root	In some conditions, if last vowel of the stem is a wide vowel (<i>geniş ünlü</i>), it is changed to a narrow vowel (<i>dar ünlü</i>). $a \rightarrow ı$ $e \rightarrow i$	<i>ağla – (i)yor – um</i> <i>ağlı – yor – um</i> <i>ağlıyorum</i> (I am crying)
Vowel reduction (<i>Ünlü daralması</i>) on suffix	In some conditions, the suffix vowel is formed as a narrow vowel, according to the last vowel of the stem. $a, ı \rightarrow ı$ $e, i \rightarrow i$ $o, u \rightarrow u$ $ö, ü \rightarrow ü$	Format: $-(y)m.ş$ <i>kazan – mış</i> (He/She had won) <i>dokun – muş</i> (He/She had touched)
Back – front vowel (<i>Kalın – ince ünlü</i>) decision for the suffix	In some conditions, the suffix vowel is formed as a back vowel or a front vowel, according to the last vowel of the stem. $a, ı, o, u \rightarrow a$ $e, i, ö, ü \rightarrow e$	Format: $-(y)s.$ <i>uyu – sa – m</i> (If I could sleep) <i>gör- se – k</i> (If we could see)

Table 6.9 Auxiliary terms to control phonetic rules

Name	Description
Last vowel of the root	Used for phonetic rules where the stem (or root) needs an alteration.
First letter of the suffix	Used for phonetic rules where the suffix needs an alteration.
Syllable count	Indicates the number of syllables in a given word. The number of vowels is equal to the syllable count in a Turkish word.

Table 6.10 Methods defined for phonetic rule controls

ID	Method Name	Method Explanation
1	BasitOlumsuzdanOlumluyaSuffixDegistirme	Used to change the suffix value with its corresponding opposite form.
2	BenzesmeKontrolu	Used for consonant assimilation controls.
3	SuffixBasinaYardimciSesEklenmesiKontrolu	Used for helper letter addition in front of the suffix controls.
4	SuffixIlkHarfSilinmesiKontrolu	Used for removal of the first letter of suffix controls.
5	KokYumusamaKontrolu	Used for consonant lenition on stem controls.
6	KokunDaralmasiKontrolu	Used for vowel reduction on root controls.
7	SuffixDaralmasiKontrolu	Used for vowel reduction on suffix controls.
8	KokunSonSesliHarfiniBulma	Used to find the last vowel of the root.
9	SuffixKalinInceKontrolu	Used to make the back – front vowel decision for the suffix.
10	YumusamayaUgrayanFiilKokleriKontrolu	Used for exceptional verb roots on consonant lenition controls.
11	GenisZamanOperasyonlari	Used for conversion operations on simple present tense, which requires syllable count and exceptional verb root controls.
12	HeceSayisiBulma	Used to get the syllable count of the stem. This is an auxiliary method mainly used for GenisZamanOperasyonlari method.
13	IsimFiilKokYardimciSesKontrolu	Additional helper letter addition controls for past tense suffixes, as they require a helper letter addition when they come after desiderative, optative and necessity suffixes, even if they are not the suffix to be translated.

Conversion operations on simple present tense suffixes needs additional controls, as there are 12 exceptional verb roots with a single syllable in Turkish that always conjugated with the format *-ır* (*-ir*, *-ur*, *-ür*). Table 6.11 lists those exceptional verb roots, which are stored under a string list defined on FSM class. All the other verb roots with a single syllable are conjugated with the format *-ar* (*-er*). If the root has more than one syllable, than it's always conjugated with the format *-ır* (*-ir*, *-ur*, *-ür*).

Table 6.11 Exceptional verb roots for simple present tense conversion

al-ır	ol-ur	öl-ür	bil-ir	bul-ur	gel-ir
kal-ır	ver-ir	var-ır	vur-ur	gör-ür	dur-ur

6.3.2 Suffix Formats

Required phonetic rule controls are correlated with different types of suffixes. As the possible changes after phonetic rule controls are known, each suffix is provided a format value, which clarifies the changeable letters, also helper letters which are used on certain occasions. Changeable letter locations are shown with a dot and helper letters are shown in parenthesis in the format of a suffix. If the format value isn't set or set but doesn't contain any special characters, it indicates that either the conversion can be done with a simple affirmative to negative or vice versa change of the suffix value at the given index, or that particular suffix is changeless and conversion operations don't affect its value. Table 6.12 lists the suffixes with their formats.

Table 6.12 Defined suffixes and their formats

ID	Suffix Name	Description	Format
1	Birinci Tekil Kişi Eki Birinci Grup	1 st person singular, 1 st group	-(y).m
2	İkinci Tekil Kişi Eki Birinci Grup	2 nd person singular, 1 st group	-s.n
3	Birinci Çoğul Kişi Eki Birinci Grup	1 st person plural, 1 st group	-(y).z
4	İkinci Çoğul Kişi Eki Birinci Grup	2 nd person plural, 1 st group	-s.n.z
5	Üçüncü Çoğul Kişi Eki Birinci Grup	3 rd person plural, 1 st group	-l.r
6	Birinci Tekil Kişi Eki İkinci Grup	1 st person singular, 2 nd group	-m
7	İkinci Tekil Kişi Eki İkinci Grup	2 nd person singular, 2 nd group	-n
8	Birinci Çoğul Kişi Eki İkinci Grup	1 st person plural, 2 nd group	-k
9	İkinci Çoğul Kişi Eki İkinci Grup	2 nd person plural, 2 nd group	-n.z

Table 6.12 continues

ID	Suffix Name	Description	Format
10	Bildirme Eki	Copula	-d.r
11	Durum Ulacı	Verbal adverb suffix, gives “as if” meaning	-
12	Bilinen Geçmiş Zaman Eki	Known past tense suffix	-(y)d.
13	Dilek Şart Kipi	Desiderative	-(y)s.
14	Öğrenilen Geçmiş Zaman Eki	Narrative past tense suffix	-(y)m.ş
15	Zaman Ulacı	Verbal adverb suffix, gives “while” meaning	-(y)ken
16	Birinci Çoğul Kişi, Dilek Kipi Çekimi	1 st person plural suffix after optative	-l.m
17	Dilek Kipi	Optative	-(y).
18	Geniş Zaman Eki	Simple present tense suffix	-r
19	Şimdiki Zaman Eki	Present continuous tense suffix	-yor
20	Gelecek Zaman Eki	Future tense suffix	-(y).c.k
21	Gerekliklik Kipi	Necessity	-m.l.
22	İkinci Çoğul Kişi Emir Kipi	Imperative (2 nd person plural)	-(y).n
23	Üçüncü Çoğul Kişi Emir Kipi	Imperative (3 rd person plural)	-s.nlar
24	Birinci Tekil Kişi Geniş Zaman Eki	Simple present tense suffix (1 st person singular)	-r.m
25	Birinci Çoğul Kişi Geniş Zaman Eki	Simple present tense suffix (1 st person plural)	-r.z
26	Mastar Eki	Infinitive suffix	-m.k
27	Kısa Mastar İyelik Çekimi Tekil	Infinitive suffix (short form), followed by 3 rd person singular possession suffix	-m.s.
28	Bulunma – Ayrılma Hal Eki	Locative – Ablative suffix	-
29	Yeterlilik Kipi Geniş Zaman	Ability/Probability suffix (Simple present tense)	-(y).bilir
30	Yeterlilik Kipi	Ability/Probability suffix	-(y).bil
31	Yeterlilik Kipi Şimdiki Zaman	Ability/probability suffix (Present continuous tense)	-(y).biliyor
32	Birinci Tekil - Birinci Çoğul Kişi Yeterlilik Kipi Geniş Zaman	Ability/Probability suffix (Simple present tense, 1 st person singular or 1 st person plural)	-(y).bilirim
33	Olumsuzluk Koşacı	Negative copula	-
34	Kısa Mastar İyelik Çekimi Çoğul	Infinitive suffix (short form), followed by 3 rd person plural possession suffix	-m.l.r.
35	Üçüncü Tekil Kişi İyelik Eki	3 rd person singular possession suffix	-(s).

Highlighted IDs on Table 6.12 points the suffixes that are not a single suffix, but a combination of suffixes gathered together for conversion phase operations.

6.3.3 Conversion from Affirmative to Negative

If the input path is affirmative, conversion operation should be towards negative. Formats for suffixes have a role, but they are not used in every condition in this conversion type. The step-by-step process for the operation is given below:

- 1) Get the appropriate suffix to translate out of the word suffix list located in the Path object.
- 2) The index of the detected affirmative value of the suffix, out of the PositiveValues list is known, so negative value with the same index out of the NegativeValues list is taken.
- 3) If necessary, related phonetic rules are applied for the suffix and stem.
- 4) If multiple suffixes exist, all other suffixes will remain as affirmative, so their format values are used to obtain the final suffix string. Giving the final forms of format values according to its suffix are described in the next section.

6.3.4 Conversion from Negative to Affirmative

If the input path is negative, conversion operation should be towards affirmative. Formats for suffixes has the main role in this conversion type. The process for the algorithm shows similarities with conversion from affirmative to negative process, but one main difference is to treat all suffixes within the suffix list equally this time, as the purpose is to obtain a positive predicate which needs all suffixes to be in positive form. Therefore, although the index of the detected negative value of the suffix, out of NegativeValues list is known, it is not single-handedly enough, and the conversion process deals with the formats of all suffixes. To give the final forms of format values, related phonetic rule controls should be made depending on the suffix type. Phonetic rules are checked via methods, which are enumerated on Table 6.10. Used methods for each suffix is sequentially listed on Table 6.13. Between methods in parenthesis, first one is used if the found stem is noun-typed, second one is used if it is verb-typed.

Table 6.13 Phonetic rule control methods used for each suffix in sequence

ID	Suffix Name	Phonetic Rule Method IDs
1	Birinci Tekil Kişi Eki Birinci Grup	3 – 5 – 7
2	İkinci Tekil Kişi Eki Birinci Grup	7
3	Birinci Çoğul Kişi Eki Birinci Grup	3 – 5 – 7
4	İkinci Çoğul Kişi Eki Birinci Grup	7
5	Üçüncü Çoğul Kişi Eki Birinci Grup	9
6	Birinci Tekil Kişi Eki İkinci Grup	1
7	İkinci Tekil Kişi Eki İkinci Grup	1
8	Birinci Çoğul Kişi Eki İkinci Grup	1
9	İkinci Çoğul Kişi Eki İkinci Grup	7
10	Bildirme Eki	2 – 7
11	Durum Ulacı	1
12	Bilinen Geçmiş Zaman Eki	(3 13) – 2 – 7
13	Dilek Şart Kipi	(3 13) – 9
14	Öğrenilen Geçmiş Zaman Eki	(3 13) – 7
15	Zaman Ulacı	1 – 3
16	Birinci Çoğul Kişi, Dilek Kipi Çekimi	7
17	Dilek Kipi	7 – 9
18	Geniş Zaman Eki	4 – 11 – 10
19	Şimdiki Zaman Eki	10 – 4 – 6 – 7
20	Gelecek Zaman Eki	10 – 9 – 3
21	Gereklilik Kipi	9 – 7
22	İkinci Çoğul Kişi Emir Kipi	3 – 7
23	Üçüncü Çoğul Kişi Emir Kipi	7 – 9
24	Birinci Tekil Kişi Geniş Zaman Eki	4 – 11 – 10 – 7
25	Birinci Çoğul Kişi Geniş Zaman Eki	4 – 11 – 10 – 7
26	Mastar Eki	9
27	Kısa Mastar İyelik Çekimi Tekil	9 – 7
28	Bulunma – Ayrılma Hal Eki	1 – 9
29	Yeterlilik Kipi Geniş Zaman	10 – 1 – 3
30	Yeterlilik Kipi	10 – 1 – 3
31	Yeterlilik Kipi Şimdiki Zaman	10 – 1 – 3
32	Birinci Tekil - Birinci Çoğul Kişi Yeterlilik Kipi Geniş Zaman	10 – 1 – 3
33	Olumsuzluk Koşacı	1
34	Kısa Mastar İyelik Çekimi Çoğul	9 – 7
35	Üçüncü Tekil Kişi İyelik Eki	1 – 3

For example, 4 different phonetic rules must be checked for the suffix type “*şimdiki zaman eki*” (present continuous tense suffix), which are possible consonant lenition on verb root controls (denoted as 10), removal of the first letter of suffix controls (as 4), vowel reduction on root controls (as 6) and vowel reduction on suffix controls (as 7).

6.3.5 Optimal Result Decision

After the elimination operations on classification phase, if there are still more than one existing path as possible solutions, a final elimination method is used to find the optimal result by checking the maximum level. As mentioned, every state has a level property, which indicates the required number of suffix parse operations to be executed on a predicate to reach that state. Accordingly, if found stem is semantically meaningful, the paths with a lower level goal state should be eliminated.

6.4 Use Case Example

A complete flow when developed model is executed on an example sentence is detailed on this section. File operations are not needed as a single sentence is handled. Sample sentence is “*Meyve suyunu içmeden önce çalkalamıyordu.*” (He/She was not shaking the fruit juice before drinking it.)

- The predicate of the sentence is found as “*çalkalamıyordu*”.
- 7 initial paths are derived from the FSM execution and one of them is directly eliminated as it is not in a goal state.
- Details of the remaining 6 paths are given below:

○ Word stem: <i>çalkalamıyor</i>	Word suffix part: <i>-du</i>
Current state: F1	Current state level: 1
○ Word stem: <i>çalkalamıyo</i>	Word suffix part: <i>-r-du</i>
Current state: F2	Current state level: 2
○ Word stem: <i>çalkalam</i>	Word suffix part: <i>-iyor-du</i>
Current state: F2	Current state level: 2
○ Word stem: <i>çalkalamı</i>	Word suffix part: <i>-yor-du</i>
Current state: F2	Current state level: 2

- Word stem: *çalkala* Word suffix part: *-mıyor-du*
Current state: F2 Current state level: 2
 - Word stem: *çalkal* Word suffix part: *-amıyor-du*
Current state: F3 Current state level: 3
- When paths with meaningless stem elimination is applied, paths numbered as 1,2,3 and 6 are eliminated and number of paths are reduced to two. Meanwhile, the classification phase is completed. Remaining paths:
 - 4) *çalkalama – yor -du* Detected stem type: Noun
Classified as: Affirmative
 - 5) *çalkala – mıyor – du* Detected stem type: Verb
Classified as: Negative
- When elimination by root type is applied, the path numbered as 4 is eliminated as the current state of this path is F2, which requires a verb-typed stem, but the detected stem is a noun. So after the path elimination operations on classification phase, a single path is remained as possible solution.
- Conversion phase begins. Only remaining path (numbered as 5) is classified as negative, so conversion operation should be executed towards affirmative.
- First suffix in the suffix part of the word is taken and program perceives that the suffix type is *SimdikiZamanEki* (present continuous tense suffix).
- Format value for *SimdikiZamanEki* is *"-.yor"*.
- Four phonetic rule control methods should be used to complete the transition centered around *SimdikiZamanEki*, which are:
 - *YumusamayaUgrayanFiilKokleriKontrolu* (10)
 - *SuffixIlkHarfSilinmesiKontrolu* (4)
 - *KokunDaralmasıKontrolu* (6)
 - *SuffixDaralmasıKontrolu* (7)
- *KokunDaralmasıKontrolu* (6) method causes a change on word stem property. Updated property is *"çalkalı"*.
- The converted affirmative predicate result is *"çalkalıyordu"*.

The conversion phase is over. The resulted sentence is *"Meyve suyunu içmeden önce çalkalıyordu."* (He/She was shaking the fruit juice before drinking it.)

Figure 6.3 shows how the system moved through the first FSM results and found the optimal result for this given example.

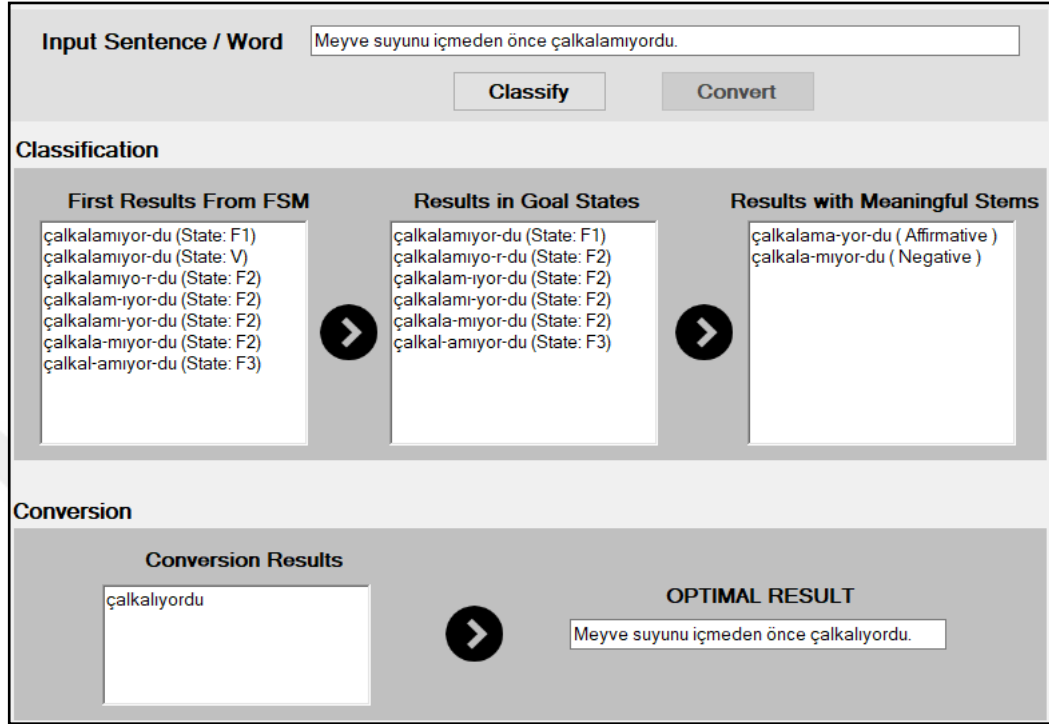


Figure 6.3 Example system output for verb polarity classification and conversion tasks

6.5 Experimental Results

Success of the proposed model is tested via experiments on actual lecture notes. The dataset with 60 documents (30 history and 30 geography), which is used on SBD module experiments is used again.

Developed model works on sentences derived from SBD model and generates a result for each sentence. This result can either be a translated sentence derived from the single remaining path (correct or incorrect), a warning text if no meaningful result is found, or a warning text if paths could not be reduced to a single one and there exists multiple candidate sentences. Besides, initial sentences without a predicate (like sentences derived by join operations on itemized text parts) are not received for consideration, as tested operations within this model are unfeasible on them. So an accuracy value, which is calculated by dividing number of correct guesses to number

of all sentences with a predicate is used for evaluation. Accuracy value is calculated for both classification and conversion phases. Also, experiments for geography and history domains are separated to allow comparisons and conclusive results are calculated by combining these two experiment sets. Evaluation metrics used on experiments are formulated below on equations 6.1 and 6.2. Combined experimental results are shown on Table 6.14.

$$\text{Classification Accuracy (\%)} = \frac{100 (\# \text{ of Correct Classifications})}{\# \text{ of Sentences with Predicate}} \quad (6.1)$$

$$\text{Conversion Accuracy (\%)} = \frac{100 (\# \text{ of Correct Conversions})}{\# \text{ of Sentences with Predicate}} \quad (6.2)$$

Table 6.14 Experiment results of verb polarity detection (classification) and conversion model

DOMAIN	# of Sentences from SBD Model	# of Sentences with Predicate	# of Correct Classifications	# of Correct Conversions	# of No Meaningful Result Cases	# of More Than a Single Result Cases	Classification Accuracy (%)	Conversion Accuracy (%)
History Documents (30)	1233	1208	1135	1117	81	5	93.96	92.47
Geography Documents (30)	1089	1080	1016	999	60	3	94.07	92.50
TOTAL	2322	2288	2151	2116	141	8	94.01	92.48
AVG	38.70	38.13	35.85	35.27	2.35	0.13		

As Table 6.14 shows, average number of sentences with predicate in the combined dataset documents is stated as 38.13 and 35.85 of them are correctly classified in terms of verb polarity, while 35.27 of them are correctly converted to the opposite polarity. When domain specific experiments are observed, average number of sentences with predicate is stated as 40.27 for history course and 36.00 for geography course documents. For history domain, 37.83 of them are correctly classified and 37.23 of them are correctly converted. For geography domain, 33.87 are correctly classified and 33.30 are correctly converted.

Experiments on history course text documents resulted in 93.96% classification accuracy and 92.47% conversion accuracy, while experiments on geography course text documents resulted in 94.07% classification accuracy and 92.50% conversion

accuracy. Combined results are stated as **94.01%** classification accuracy and **92.48%** conversion accuracy.

Results show that there is not a considerable difference between the success rates of history and geography domains. Among the history documents, average number of cases where no meaningful result is found is 2.70 and more than a single result is found is 0.17. Numbers are calculated as 2.00 and 0.10 respectively for geography documents. When more than a single result is found, that means the system is unable to resolve the ambiguity as there exists two different meaningful stems followed by appropriate suffix lists for each. Figure 6.4 shows an example condition, where a single path could not be gathered as both candidate verbal stems (“*sür*” (drive, lead, continue) and “*sürü*” (drag, herd)) are meaningful and appended suffixes lead both of them to a proper goal state in FSM. This is an example condition which exceeds the limits of morphology and entered the scope of semantics.

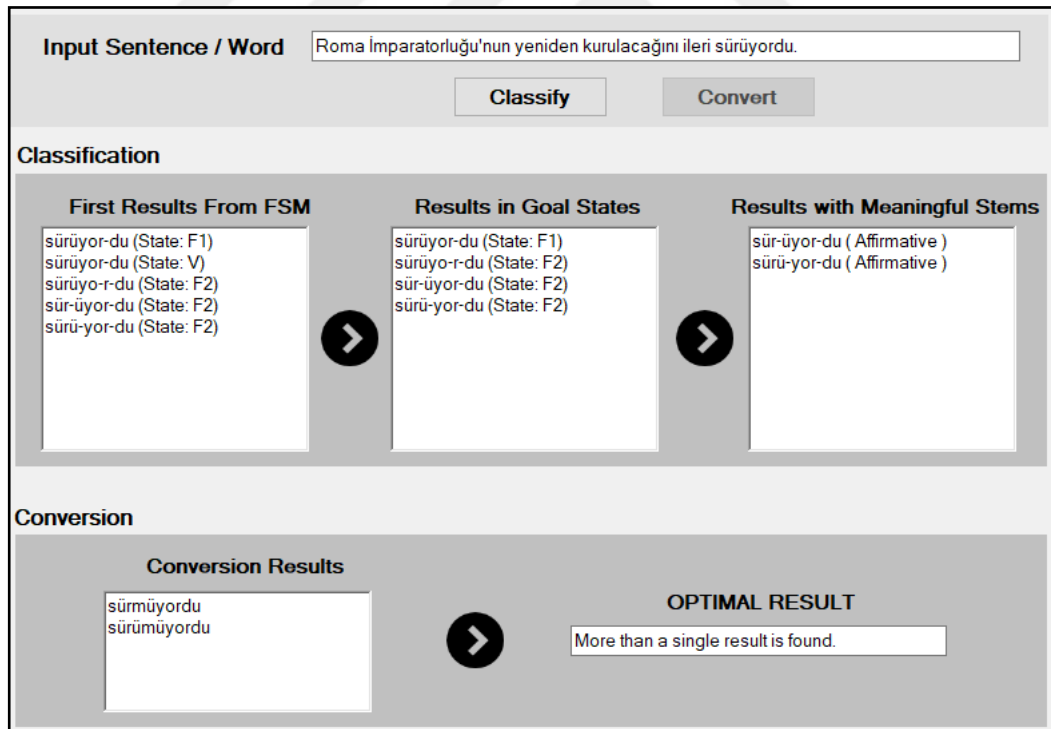


Figure 6.4 Example case where multiple results remain after FSM execution and path elimination

CHAPTER SEVEN

NAMED ENTITY RECOGNITION MODEL TO BUILD A GLOSSARY OF TERMS STRUCTURE

7.1 Overview

Proposed named entity recognition (NER) model is an information extraction software developed for educational purposes and specialized for Turkish lecture notes within geography and history domains. Primary goal of the model is to detect named entities from the context of input text documents with high accuracy. Using qualified named entities among the detected, building a steady and satisfying glossary of terms structure for history and geography domains is defined as the next step. Later on, this structure is used to support the test preparation process.

Implemented NER model exploits a rule-based approach. It takes a text document as input and returns detected named entities with their types as output. System is developed to work on sentences, so sentence boundary detection (SBD) model is executed on input text file first. This operation provides sentences and headings of input text for NER system usage. Therefore, the success of NER model also depends on the success of SBD model.

Each sentence is handed to NER respectively and processed with the tokenizer, lexical model and contextual model. These three sub-models prepare given sentence by providing informative labels. Finally, the recognizer model is executed and the sentence with labeled tokens is analyzed to detect named entities. Figure 7.1 shows a representation of the proposed framework.

Building the glossary of terms task is handled after the execution of NER model on numerous text-based lecture notes within history and geography documents. Resulted named entities of different types are observed and the ones that might be used for an examination process are specified. A more specific categorization for selected terms is also done on this stage to combine interchangeable terms with each other.

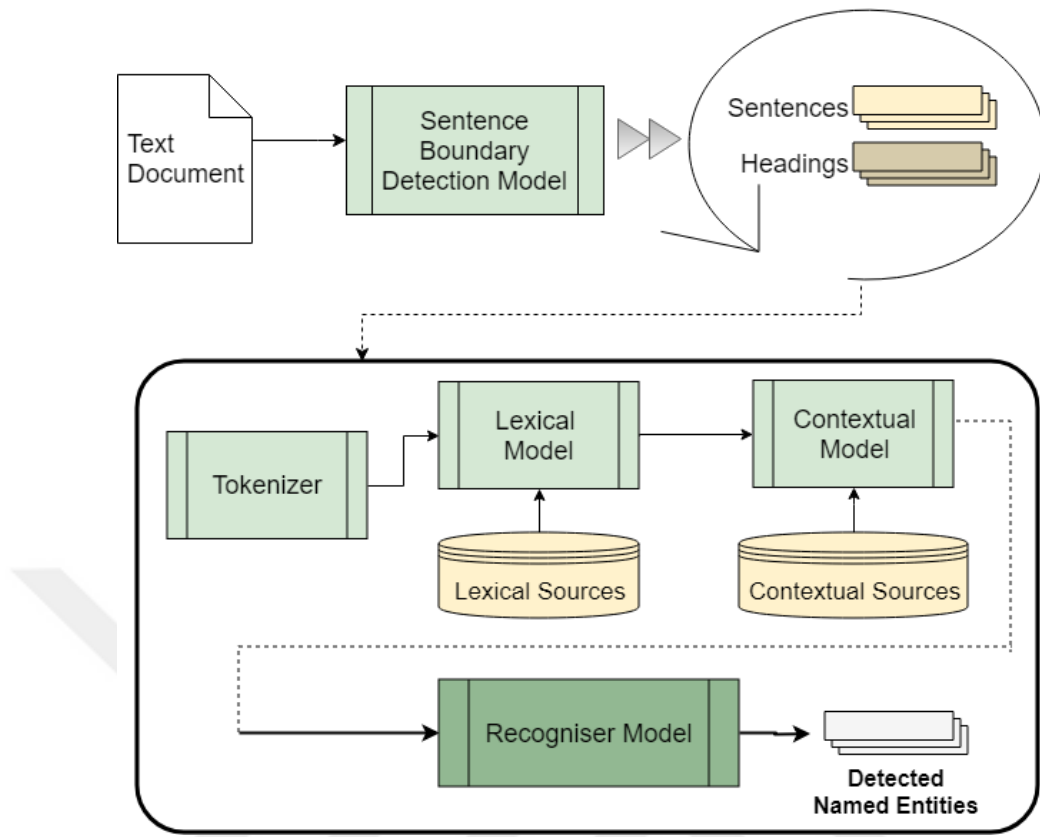


Figure 7.1 Flow diagram of the proposed named entity recognition model

7.2 Tokenizer and Tokens

Derived sentences of the input text file are first processed by tokenizer. Tokenizer scans through the input sentence and detects word boundaries and punctuation marks to get the list of tokens. A token might indicate a complete word, a punctuation mark or a morpheme after a punctuation mark. Tokens of a sentence are stored in a double linked list structure as a Token class object is designed to hold the information of previous and next tokens. A token object also holds a list of boolean variables that indicate states (labels). Labeling a token provides useful background information to be used while detecting named entities. Tokenizer applies the initial labeling on collected tokens. Considering the system requirements on further stages, 15 tokenizer labels divided to four different categories are defined. Case, numeric, punctuation and location information are provided by labeling on this stage. Labels used by tokenizer are shown on Table 7.1.

Table 7.1 Categorized tokenizer labels

Case Information	Numeric Information	Punctuation Information	Location Information
SW_CAPITAL	NUM	PUNCT_APOSTR	BEFORE_APOST
ALL_CAPITAL	ROMAN_NUM	PUNCT_OTHER_MID	
	ORD_NUM	PUNCT_OTHER_END	
	DAY_NUM		AFTER_APOST
EW_DOT	MONTH_NUM	PERCT	
	YEAR_NUM		

- SW_CAPITAL: Indicates whether the token text starts with a capital letter or not.
- ALL_CAPITAL: Indicates whether all characters of the token text are capitalized or not.
- EW_DOT: Indicates whether the last character of the token is a dot or not.
- NUM: If set to true, indicates that the token text denotes a numeric value.
- ROMAN_NUM: If set to true, indicates that the token text denotes a roman number.
- ORD_NUM: If set to true, indicates that the token text denotes an ordinal number.
- DAY_NUM: If set to true, indicates that the token holds a numeric value in [1,31] range.
- MONTH_NUM: If set to true, indicates that the token holds a numeric value in [1,12] range.
- YEAR_NUM: If set to true, indicates that the token holds a numeric value in [100, 5500] range.
- PUNCT_APOSTR: Indicates whether the token text is an apostrophe character or not.
- PUNCT_OTHER_MID: Indicates whether the token text holds a punctuation mark used in the middle of a sentence, like comma, semi colon, parenthesis etc.
- PUNCT_OTHER_END: Indicates whether the token text holds a sentence ending punctuation mark (except dot) or not.
- PERCT: Indicates whether the token text is a percentage sign or not.
- BEFORE_APOST: If set to true on a token, points that next token is an apostrophe.

- **AFTER_APOST:** If set to true on a token, points that previous token is an apostrophe.

7.3 Generated Sources for Lookup Operations

7.3.1 Lexical Model Sources

Lexical and contextual models are used to label tokens with additional information using the generated lexicon structures. Lexicons used by lexical model indicates possible proper names (of a person or a location-region) except the auxiliary list which contains Turkish conjunctions. Lexical model sources are detailed below:

- **TR_FirstNames:** Stores Turkish first names based on a database that contains Turkish Language Association (TDK) person names dictionary terms. Initial list holds 9699 elements, but the number is reduced to 9619 after some elimination, which will be detailed on Section 7.3.1.1.
- **TR_CommonSurnames:** Stores a comprehensive list of Turkish surnames which are extracted from Wikipedia lists for Turkish actors – actresses, Turkish politicians (from 20th and 21st centuries), Turkish writers and Turkish commanders in Turkish War of Independence. Multiple occurrences of the same person (for example a politician who has served on both 20th and 21st centuries) and the duplicates of frequent surnames are eliminated. Final list contains 3039 elements.
- **FRGN_FirstNames:** Stores a list of foreign (not Turkish) first names, derived from a wiki list projected to be expanded by user provided entries, published on *ranker.com* (“The Most Influential People of All Time”, n.d.). This list consists of a total 2762 people in a wide spectrum like scientists, politicians, artists, athletes, philosophers etc. from different countries. Data is extracted as an XML file, then normalized to get plain lists of first names, surnames and mid names. Normalization phase includes the removal of prepositions or articles like “of, the”, ordinal numbers, roman numbers and words that indicate a title or a nickname (like “St, Holy, Crazy, King, Queen, Baron, Prince, Princess”). Duplicate occurrences of a name are also excluded. Final list contains 1489 elements.

- **FRGN_CommonSurnames:** Stores a list of foreign surnames. Foreign surnames and mid names are also derived from the source list from *ranker.com*. Final list contains 1864 elements.
- **FRGN_MidNames:** Stores a list of foreign mid names like “de, von, bin” or shortened forms which is an initial upper-case letter trailed by a dot. Final list contains 34 elements.
- **Countries:** Stores the names of 193 member states of United Nations (UN), states consisting in these members (like England, Wales, Scotland, Northern Ireland) and self-governing states (like Puerto Rico, Virgin Islands, New Caledonia). Palestine, Taiwan and TRNC (Turkish Republic of Northern Cyprus) are the other states included. Some former country names that are likely to occur in historical texts (like Yugoslavia, USSR) are also included. Final list contains 257 elements.
- **TR_Cities:** Stores the names of 81 cities of Turkey and common different usages for them (like Afyon for Afyonkarahisar). Final list contains 86 elements.
- **TR_Districts:** Stores the names of districts of Turkey. Initial list holds 984 elements but after districts with same names and central districts named after their inclusive city are eliminated, final list contains 897 elements.
- **FRGN_StatesCities:** Stores the names of capital cities of all countries and states-cities with high population, or historical and touristic significance. Cities that are named after their countries are excluded and the final list contains 380 elements.
- **GeographicRegions:** Stores the names of continents or well-known geographic regions. The list contains 22 elements.
- **Conjunctions:** Stores conjunctions used in Turkish language. This auxiliary list is used to detect conjunction usage at the beginning of a sentence to avoid misleading named entity (NE) detections.

7.3.1.1 Final Exclusions from Lexical Sources

Initial list taken from *ranker.com* contains some Turkish people like Mustafa Kemal Atatürk, Halide Edip Adıvar, Orhan Veli Kanık, Yunus Emre. This led some intersection between Turkish name lists and foreign name lists. 29 mutual words are detected between TR_FirstNames and FRGN_FirstNames lists, while 13 mutual words are detected between TR_CommonSurnames and FRGN_CommonSurnames

lists. Leaving some of them on both lists are considered appropriate but some of them are excluded from one of the lists, as detailed below:

- Words like “Abdullah, Selma, Selman, Zakir” etc. are left on both lists.
- Words like “Edip, Evliya, Halide, Hamdi, Kemal, Mustafa, Orhan, Yunus, Ziya” etc. are excluded from FRGN_FirstNames lists.
- Words like “Adam, Alan, Boy, Sun, San” etc. are excluded from TR_FirstNames list.
- Words like “Adivar, Çelebi, Emre, Kanık, Pamuk, Atatürk, Tanpınar” etc. are excluded from FRGN_CommonSurnames list.
- Words like “Bradley, Reynaud, Spence” are excluded from TR_CommonSurnames list. These elements came from the names of Turkish people of foreign origin or married to a foreign person.
- In lexical sources, there also exists some overlap with contextual model sources. These overlapping words are excluded from lexical sources to give them their final forms.

7.3.2 Contextual Model Sources

Source lists used by contextual model indicates possible neighbor expressions for proper names. These expressions might be in the NE text or not, their case information is mostly the criteria looked for this decision. Contextual model sources are detailed below:

- **Before Person lists:** Stores words or word groups that might come before a person name. Four lists are used for this purpose. Lists include profession titles like “*Lord, Gazi, Albay*” (Lord, Veteran, Colonel), honorifics like “*Bay, Bayan, Madam*” (Mister, Missis, Madam), abbreviations like “*Asb., Prof., Yzb.*” (Sgt., Prof., Capt.) and mid-expressions like “*komutanı, padişahı, valisi*” (commander of, sultan of, governor of).
- **After Person:** Stores profession titles in Turkish like “*Efendi, Hatun, Han, Paşa*” that possibly come after a person name.

- **After State or Country lists:** Stores words or word groups that might come after a state or country name. Two lists are used for this purpose. One list includes ending expressions like “*Krallığı, Cumhuriyeti*” (Kingdom, Republic), other includes mid-expression like “*başbakanı, halifesi, imparatoru*” (prime minister of, emperor of, khalifa of).
- **After Location:** Stores words or word groups that might come after a location name other than a state or country. The list includes expressions like “*belediye başkanı, Bölgesi, valisi*” (mayor of, Region, governor of).
- **After Organization:** The list includes expressions like “*Derneği, Meclisi, Kurumu*” (Association, Council, Institution).
- **After Geographical Formations:** The list includes expressions like “*Gölü, Dağı, Irmağı*” (Lake, Mountain, River). There also exists a list which holds possible expressions that a geographical formation ends with in Turkish like “*ırmak, dağlar*” etc.
- **After Geographical Events:** The list includes expressions like “*Depremi, Yangını*” (Earthquake, Fire).
- **After Historic Events:** The list includes expressions like “*Savaşı, Devrimi, İsyanı*” (War, Revolution, Riot).
- **After Historic Buildings:** The list includes expressions like “*Sarayı, Köprüsü*” (Palace, Bridge).
- **Months:** Holds the names of the months.

7.4 Labeling by Lexical and Contextual Models

Tokenizer parses a sentence, generates tokens and initially labels them. Unlike tokenizer, lexical and contextual models don’t label tokens one by one, as some lexicon terms might contain multiple words. Thus, tokens are passed to these models with n-grams. Initial token window width is defined as 4 and it decreases on every iteration until it reaches to zero. Multi-word lexicon terms are not missed and labeled correctly this way. Table 7.2 shows how the n-gram search patterns are modeled on a sentence of 7 tokens.

Table 7.2 Search patterns of a 7-token sentence for n-gram lexicon lookups

N Value	Search Patterns
4	1234 – 2345 – 3456 – 4567
3	123 – 234 – 345 – 456 – 567
2	12 – 23 – 34 – 45 – 56 – 67
1	1 – 2 – 3 – 4 – 5 – 6 – 7

Tokens are labeled via n-gram lexicon lookups in lexical and contextual models to get their final forms before the execution of recognizer model. Table 7.3 shows the labels used in lexical and contextual models.

Table 7.3 Lexical (L) and contextual (C) model labels

Model	Label Name	Description
L	LEX_TR_FN	Lexical term, Turkish first name
L	LEX_TR_LN	Lexical term, Turkish last name
L	LEX_FRGN_FN	Lexical term, foreign first name
L	LEX_FRGN_MN	Lexical term, foreign mid-name
L	LEX_FRGN_LN	Lexical term, foreign last name
L	LEX_CTRY	Lexical term, country name
L	LEX_TR_CITY	Lexical term, Turkish city name
L	LEX_TR_DIST	Lexical term, Turkish district name
L	LEX_FRGN_CITY	Lexical term, foreign city name
L	CONJ_SWC	Conjunction that starts with capital
L	NOT_LEX_SWC	Not a lexical term but starts with capital
C	B_PERSON	Before person expression
C	A_PERSON	After person expression
C	A_LOC_CTRY	After location – country expression
C	A_LOC_OTH	After location (other) expression
C	A_ORG	After organization expression
C	A_HIST_BLDG	After historic building expression
C	A_HIST_EVNT	After historic event expression
C	A_GEO_FORM	After geographic formation expression
C	A_GEO_EVNT	After geographic event expression
C	EW_GEO_FORM	Indicates a possible geographic formation with its ending
C	MONTH_NAME	Indicates a month name

Figure 7.2 shows a use case example of tokenization and token labeling with three different models on the sentence “*Dünya’da 23 Eylül günü, Türkiye Cumhuriyeti’nde ve tüm Kuzey Yarım Küre’de sonbahar başlar.*” (On the day of 23 September in the world, it is the beginning of autumn in Turkey and the whole Northern Hemisphere.). Token labels from different models are shown with different colors.

Dünya’da 23 Eylül günü, Türkiye Cumhuriyeti’nde ve tüm Kuzey Yarım Küre’de sonbahar başlar.			Tokenize and Label	
	BLACK: Labels from Tokenization	GREEN: Labels from Lexical Modal	BLUE: Labels from Contextual Modal	
(1) Dünya	STARTS_WITH_CAPITAL			
(2) ,	PUNCT_APOSTR			
(3) da	AFTER_APOSTR	FRGN_MIDNAME		
(4) 23	NUMERIC	DAY_NUM		
(5) Eylül	STARTS_WITH_CAPITAL	MONTH_NAME		
(6) günü				
(7) ,	PUNCT_OTHER_MID			
(8) Türkiye	STARTS_WITH_CAPITAL	COUNTRY_REGION		
(9) Cumhuriyeti	STARTS_WITH_CAPITAL	BEFORE_APOSTR	AFTER_LOC_COUNTRY	
(10) ,	PUNCT_APOSTR			
(11) nde	AFTER_APOSTR			
(12) ve				
(13) tüm				
(14) Kuzey	STARTS_WITH_CAPITAL			
(15) Yarım	STARTS_WITH_CAPITAL			
(16) Küre	STARTS_WITH_CAPITAL	BEFORE_APOSTR	TR_DISTRICT	AFTER_GEO_FORM
(17) ,	PUNCT_APOSTR			
(18) de	AFTER_APOSTR	FRGN_MIDNAME		
(19) sonbahar				
(20) başlar				

Figure 7.2 Example system output after tokenization and token labeling on an input sentence

7.5 Named Entities and Recognizer Model

As developed NER system is specialized for lecture notes in the scope of history and geography courses, extent of a NE is adjusted to meet the requirements. 13 NE types are defined, which are explained on Table 7.4.

After token derivation and labeling is completed, recognizer is executed to find out named entities. System can both be tested on a single sentence or a complete text document. Figure 7.3 shows a use case example where the system is tested with the input sentence “*Bornova Anadolu Lisesi ve İzmir Atatürk Lisesi öğrencileri, Cumhuriyet Bayramı’nı kutlamak için Gündoğdu Meydanı’nda toplandı.*” (Students of Bornova Anatolian High School and İzmir Atatürk High School are gathered in Gündoğdu Square to celebrate Republic Day.). Execution resulted in four NE detections. Tokens “*Bornova, İzmir, Atatürk, Gündoğdu*” are all lexicon terms and might be named entities on their own in different sentences. On the example though,

these terms are correctly found to be parts of longer named entities. System is designed to consider the container named entities instead of single lexicon terms in such circumstances.

Table 7.4 Defined NE types with their explanations

Label Name	Description
Person Turkish	Indicates a Turkish person name
Person Foreign	Indicates a foreign (not Turkish) person name
Location State - Country	Indicates a country, state, continent or geographic region
Location Other	Indicates a city or district
Historic Term Building	Indicates a historic building or structure
Historic Term Event	Indicates a historical event
Geographic Term Formation	Indicates a specific geographical formation
Geographic Term Event	Indicates a specific geographical event such as a natural disaster
Organization	Indicates an organization within a wide range of fields (politics, education, military, media, law, medical etc.)
Percentage	Indicates a percentage or fraction expression
Date	Indicates a single date expression in multiple formats, or a date range expression
Date or Number	Indicates a clock expression or a numeric value below 1200 or above 2000.
Other	Indicates a detected NE which is not classified as one of the distinctive types.

Bornova Anadolu Lisesi ve İzmir Atatürk Lisesi öğrencileri, Cumhuriyet Bayramı'nı kutlamak için Gündoğdu Meydanı'nda toplandı.

Tokenize and Label

BLACK: Labels from Tokenization GREEN: Labels from Lexical Modal BLUE: Labels from Contextual Modal

(1) Bornova	STARTS_WITH_CAPITAL	TR_DISTRICT	
(2) Anadolu	STARTS_WITH_CAPITAL		
(3) Lisesi	STARTS_WITH_CAPITAL	AFTER_ORG	
(4) ve			
(5) İzmir	STARTS_WITH_CAPITAL	TR_CITY	
(6) Atatürk	STARTS_WITH_CAPITAL	TR_FIRSTNAME	TR_LASTNAME
(7) Lisesi	STARTS_WITH_CAPITAL	AFTER_ORG	
(8) öğrencileri			
(9) ,	PUNCT_OTHER_MID		
(10) Cumhuriyet	STARTS_WITH_CAPITAL		
(11) Bayramı	STARTS_WITH_CAPITAL	BEFORE_APOSTR	AFTER_HIST_EVENT
(12) .	PUNCT_APOSTR		
(13) nı	AFTER_APOSTR		
(14) kutlamak			
(15) için			
(16) Gündoğdu	STARTS_WITH_CAPITAL	TR_FIRSTNAME	TR_LASTNAME
(17) Meydanı	STARTS_WITH_CAPITAL	BEFORE_APOSTR	AFTER_HIST_BLDG
(18) ,	PUNCT_APOSTR		
(19) nda	AFTER_APOSTR		
(20) toplandı			

Detected Named Entities

(1) Bornova Anadolu Lisesi ORGANIZATION
(2) İzmir Atatürk Lisesi ORGANIZATION
(3) Cumhuriyet Bayramı HISTORIC_TERM_EVENT
(4) Gündoğdu Meydanı HISTORIC_TERM_BUILDING

Figure 7.3 Example system output that shows named entity detections on an input sentence

7.6 Building Glossary of Terms Structure

7.6.1 NER Execution on Complete Dataset

To construct glossary of terms structure, developed NER model is executed on the complete dataset of 1200 documents (600 geography and 600 history) which is introduced on Section 4.1. At the first stage, repetitive occurrences are not eliminated to derive suggestive results about the distribution of named entities between different domains and the total counts. This led to 42442 initial named entities, which makes an average 35.37 per document.

When the resulted named entities are observed, country names are specified to be among the most homogenously distributed terms. For example, the term “*Hindistan*” (India) is seen on 70 different documents, 30 of them are in geography domain while 40 of them are in history domain. On the other hand, European and Asian countries are more frequently mentioned in history documents, while African, South American and Australian countries are mostly seen in geography documents. Former country names like Yugoslavia and USSR are nearly always mentioned in history documents. Making generalized statements for terms where country and continent names are used with a direction is not feasible, as some of them like “*Güney Asya*” (South Asia), “*Güney Afrika*” (South Africa) are mostly seen on geography documents, while some terms like “*Doğu Avrupa*” (Eastern Europe) are mostly seen on history documents.

When distributions of city and district names are observed, it can be said that their occurrence in geography documents is more frequent, but making a generalization is not possible. Especially, some of the foreign city names like “*Beyrut*” (Beirut), “*Bağdat*” (Baghdad), “*Gazze*” (Gaza), “*Hiroşima*” (Hiroshima), “*Kudüs*” (Jerusalem), “*Moskova*” (Moscow) only exist in history document. On the other hand, the most homogenously distributed terms mostly seem to be Turkish city names like “*Şanlıurfa*” (17 G – 17 H), “*Bitlis*” (17 G – 17 H), “*Karaman*” (7 G – 7 H), “*Çanakkale*” (35 G – 36 H), “*Erzurum*” (46 G – 48 T), “*Diyarbakır*” (22 G – 24 H), “*Kastamonu*” (11 G – 12 H), “*İzmir*” (45 G – 51 H), where ‘G’ stands for geography and ‘H’ stands for history documents.

Another observation is that, some of the Turkish district names in TR_Districts lexicon might lead to misleading named entity type detections as they can refer to different types. For example district names like “*Eyüp, Fatih, İnönü, Çelebi, Selim*” are prone to be used as person names, while “*Perşembe* (Thursday), *Pazar* (Sunday), *Aralık* (December)” mostly indicate name of a day or month, or terms like “*Bor* (Boron), *Bozkır* (Steppe), *Çay* (Tea), *Çeltik* (Paddy), *Maden* (Mine)” mostly indicate a geographical term like an agricultural product, a vegetation cover or a mineral type. 46 district names in TR_Districts lexicon are stated to be misleading and are not included in glossary of terms structure.

After elimination of duplicate and irrelevant named entities, distinctive named entities that might be serviceable for an exam preparation process are specified. This resulted in a total of 3939 primary terms and 921 synonym terms. Each synonym term represents a different spelling variation that actually indicates the same entity with a primary term.

7.6.2 Fine Grained Categorization

13 NE types which are defined within NER model provides an initial classification of terms. On the glossary of terms structure, Date and Date or Number types or combined as a single Date type, while Percentage type is excluded. Person Turkish type is renamed as Person Group 1 to extend the initial scope with people from communities that have considerable cultural affinity and historical interactions with Turkish communities, like Mongols and Huns. Person Foreign type is also renamed as Person Group 2 in this direction. An additional <Generic> type is included for some more general terms that are mostly observed on heading texts. After all, 12 NE labels for coarse grained classification is defined.

To increase the question quality and provide a more specific classification, 311 fine grained categories are defined within the outer 12 labels. With this approach, more related terms, which are interchangeable in true-false and multiple choice questions are gathered together. For example, if this second level categorization wasn't applied, different Person Group 1 terms that indicate an Ottoman sultan, a member of

parliament in Turkish Republic era and a minstrel would all be in consideration as a candidate term for the same question, which is not an optimal circumstance. Table 7.5 shows categories with most terms for each 12 NE label.

Table 7.5 Categories with most terms for each coarse grained named entity labels

NE Label	Categories
<Generic>	<History Term> → 18 terms <Geography Term> → 3 terms
Date	Turkish War of Independence Era Event → 156 terms World History Event – 20 th Century (After 1950) → 88 terms First World War Era Event → 78 terms
Location Other	Marmara Region District → 62 terms Aegean Region District → 56 terms Black Sea Region District → 56 terms
Location State - Country	Middle East and North Africa Countries → 22 terms Geographical Area of a Continent → 20 terms Second Period Anatolian Beylics → 19 terms
Person Group 1	Ottoman Sultan → 34 terms Ottoman Grand Vizier → 32 terms Ottoman and Turkish Republic Era Soldier and Politician → 29 terms
Person Group 2	European Scientist (19 th Century and After) → 27 terms American Scientist (19 th Century and After) → 26 terms Medieval Era European Emperor and Military Leader → 24 terms
Organization	World Economical – Political Community → 31 terms Turkish War of Independence Era Helpful Union → 18 terms Ottoman Era Military Organization → 17 terms
Geographic Term Formation	Turkey Lake Name → 52 terms Turkey Mountain Chain Name → 34 terms Mountain Name → 27 terms
Geographic Term Event	Climate Type → 8 terms Geological Period → 5 terms
Historic Term Building	Mosque Name → 24 terms Castle Name → 19 terms Bridge Name (Turkey and Ottoman) → 11 terms
Historic Term Event	Historical Treaty Name (20 th Century) → 24 terms Historical Treaty Name (Before 20 th Century) → 29 terms Historical Congress Name → 25 terms
Other	Religion and Sect Name → 16 terms Book Name → 10 terms Language Name → 8 terms

Fine grained categorization approach resulted in an average of 12.66 terms per category. Table 7.6 shows the distribution of categories based on the number of terms they contain. [5,10) is identified to be the most frequent range for number of terms with 83 categories.

Table 7.6 Distribution of categories based on the number of terms they contain

Number of Terms (Within a Range)	Number of Categories
[2,5)	78
[5,10)	83
[10,15)	60
[15,20)	35
[20,25)	19
[25,30)	18
[30,50)	11
[50,100)	6
100 and more	1

Additional to the primary 311 categories, 3 exceptional categories are defined in the scope of the NE labels. A term can both have a primary and exceptional category, but exceptional categories override the primary category when encountered. For example, exceptional category “District Name Which is Also a Lowland” is defined under Location Other type and contains 9 terms (like “*Pamukova, Karlhova, Taşova*”) with a variety of primary categories.

7.6.3 Database Model

To store the specified terms within geography and history domains and provide suggestive information about them, a database model is constructed. Data is stored in 4 different tables:

- NamedEntityType
- Category
- Term
- SynonymTerm

Database diagram of the model is given in Figure 7.4.

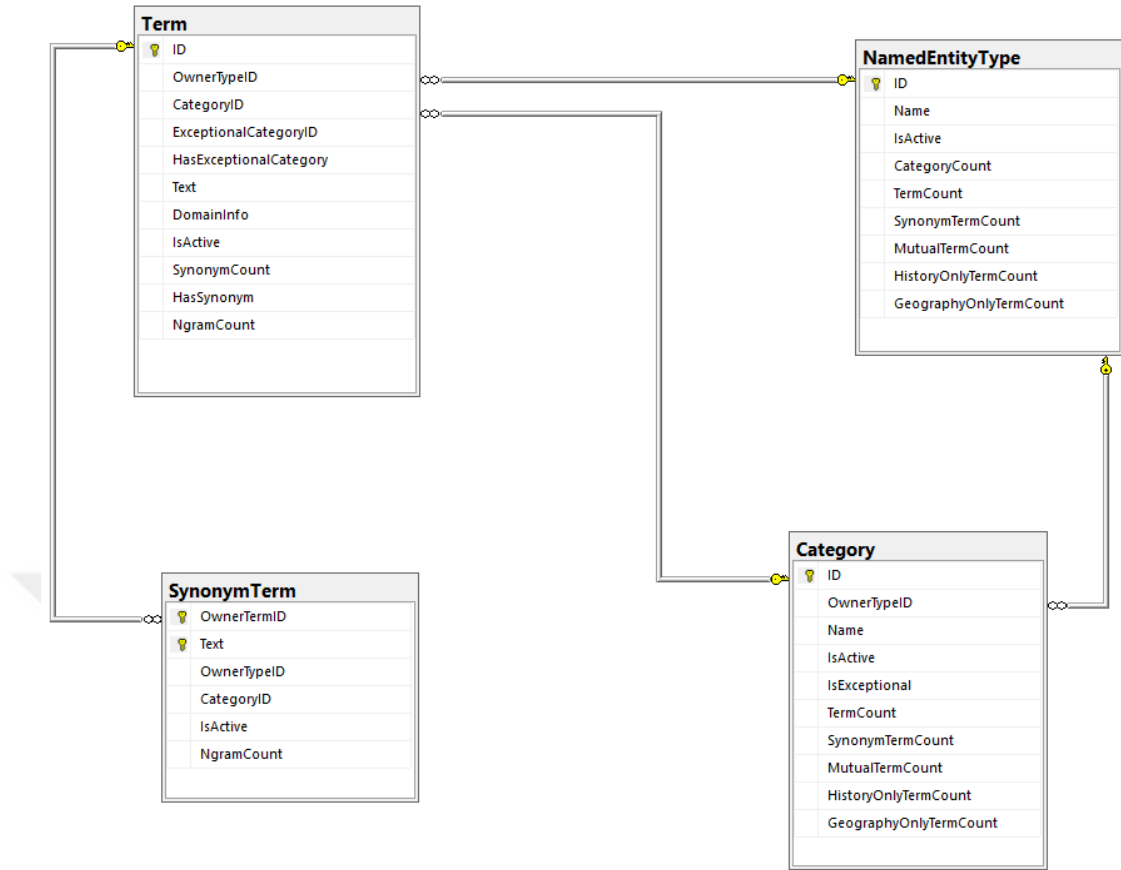


Figure 7.4 Diagram of the GlossaryOfTerms database

7.6.3.1 NamedEntityType Table

Information about the named entity labels used for coarse grained classification of terms is stored under NamedEntityType table. 9 fields are defined within this table:

- ID: Unique index value for the NE type
- Name: Explanatory name of the NE type
- IsActive: A bit field which indicates whether the NE type is enabled for lookup search operations or not. If set to 0 (false), terms of this NE type are excluded from the initial list of terms to be used on test preparation process.
- CategoryCount: Indicates total number of categories defined within this NE type.
- TermCount: Indicates total number of terms defined within this NE type.

- **SynonymTermCount:** Indicates total number of synonym terms defined within this NE type.
- **MutualTermCount:** Indicates total number of terms within this NE type that are encountered on both history and geography documents.
- **HistoryOnlyTermCount:** Indicates total number of terms within this NE type that are encountered only on history documents.
- **GeographyOnlyTermCount:** Indicates total number of terms within this NE type that are encountered only on geography documents.

Figure 7.5 shows data stored in this table.

ID	Name	IsActive	CategoryCount	TermCount	SynonymTermCount	MutualTermCount	HistoryOnlyTermCount	GeographyOnlyTermCount
1	GENERIC	1	2	28	6	5	19	4
2	DATE	1	17	581	0	6	536	39
3	LOCATION_OTHER	1	42	615	28	196	149	270
4	LOCATION_STATE_COUNTRY	1	47	429	132	116	242	71
5	PERSON_GROUP1	1	42	598	330	7	591	0
6	PERSON_GROUP2	1	19	288	137	2	270	16
7	ORGANIZATION	1	36	308	83	3	295	10
8	GEOGRAPHIC_TERM_FORMATION	1	43	496	70	42	57	397
9	GEOGRAPHIC_TERM_EVENT	1	5	21	15	2	4	15
10	HISTORIC_TERM_BUILDING	1	15	115	11	3	88	24
11	HISTORIC_TERM_EVENT	1	33	385	93	2	382	1
12	OTHER	1	13	75	16	12	50	13

Figure 7.5 Data stored in NamedEntityType table

7.6.3.2 Category Table

Information about the category labels used for fine grained classification of terms is stored under Category table. 10 fields are defined within this table:

- **ID:** Unique index value for the category entity.
- **OwnerTypeID:** Index value of the container NE type of the category.
- **Name:** Explanatory name of the category
- **IsActive:** A bit field which indicates whether the category is enabled for lookup search operations or not. If set to 0 (false), terms of this category are excluded from the initial list of terms to be used on test preparation process.
- **IsExceptional:** A bit field which indicates the category is an exceptional type or not.
- **TermCount:** Indicates total number of terms defined within this category.

- **SynonymTermCount:** Indicates total number of synonym terms defined within this category.
- **MutualTermCount:** Indicates total number of terms within this category that are encountered on both history and geography documents.
- **HistoryOnlyTermCount:** Indicates total number of terms within this category that are encountered only on history documents.
- **GeographyOnlyTermCount:** Indicates total number of terms within this category that are encountered only on geography documents.

Figure 7.6 shows sample data stored in this table.

ID	OwnerType...	Name	IsActive	IsExceptional	TermCount	SynonymTerm...	MutualTerm...	HistoryOnly...	GeographyOnly...
165	6	Avrupalı Kağıt	1	0	11	4	1	8	2
166	6	Orta Doğu Lider	1	0	10	7	1	9	0
167	6	İlkçağ Hükümdar	1	0	20	11	0	20	0
168	6	İlkçağ Bilim İnsanı ve Siyasetçi	1	0	10	7	0	10	0
169	6	Ortaçağ Avrupa Sanatçı	1	0	5	2	0	5	0
170	7	Osmanlı Dönemi Siyasi Parti	1	0	10	8	0	10	0
171	7	Türkiye Cumhuriyeti Dönemi Siyasi Parti	1	0	10	3	0	10	0
172	7	Yabancı Devlet Siyasi Parti	1	0	7	3	0	7	0
173	7	Kurtuluş Savaşı Dönemi Zararlı Cemiyet	1	0	12	4	0	12	0
174	7	Osmanlı ve Türkiye Cumhuriyeti Siyasi Dönem	1	0	8	0	0	8	0
175	7	Dünya Ekonomik - Siyasi Topluluk	1	0	32	10	2	29	1
176	7	Kurtuluş Savaşı Dönemi Yararlı Cemiyet	1	0	18	10	0	18	0
177	7	Osmanlı Dönemi Cemiyet - Demek	1	0	6	3	0	6	0

Figure 7.6 Sample data stored in Category table

7.6.3.3 Term Table

Information about the primary terms specified after NER operations is stored under Term table. 11 fields are defined within this table:

- **ID:** Unique index value for the term entity.
- **OwnerTypeID:** Index value of the container NE type of the term.
- **CategoryID:** Index value of the container category of the term.
- **ExceptionalCategoryID:** Index value of the exceptional category of the term, if one is assigned. NULL is the default value of this field if an exceptional category is not assigned to the term.
- **HasExceptionalCategory:** A bit field which indicates whether an exceptional category is assigned to the term or not.
- **Text:** Complete textual representation of the term.

- **DomainInfo:** Nchar(1) field which provides insight about the distribution of the term within the domains. ‘H’ is assigned if the term only exists on history documents, ‘G’ is assigned if the term only exists on geography documents, ‘M’ is assigned if the term is encountered on both history and geography documents.
- **IsActive:** A bit field which indicates whether the term is enabled for lookup search operations or not. If set to 0 (false), that particular term is excluded from the initial list of terms to be used on test preparation process.
- **SynonymCount:** Indicates total number of synonym terms related with that particular term.
- **HasSynonym:** A bit field which indicates whether the term has a synonym term or not.
- **NgramCount:** Indicates total number of n-grams of the term, which is calculated by adding 1 to the number of whitespaces within the Text field of the term.

Figure 7.7 shows sample data stored in this table.

ID	OwnerTypeID	CategoryID	Except...	HasExcept...	Text	DomainInfo	IsActive	SynonymCount	HasSynonym	NgramCount
1344	4	65	NULL	0	Vatikan	M	1	0	0	1
1345	4	65	NULL	0	Yunanistan	M	1	0	0	1
1346	4	72	NULL	0	Birmania	H	1	2	1	1
1347	4	72	NULL	0	Endonezya	M	1	0	0	1
1348	4	72	79	1	Filipinler	G	1	0	0	1
1349	4	72	NULL	0	Kambocya	H	1	0	0	1
1350	4	72	79	1	Kiribati	G	1	0	0	1
1351	4	72	79	1	Malezya	G	1	0	0	1
1352	4	72	NULL	0	Singapur	G	1	0	0	1
1353	4	72	NULL	0	Tayland	G	1	0	0	1
1354	4	72	NULL	0	Vietnam	H	1	0	0	1
1355	4	72	NULL	0	Yeni Zelenda	M	1	0	0	2
1356	4	89	NULL	0	Büyük Bulgar Hanlığı	H	1	1	1	3
1357	4	89	NULL	0	Çağatay Hanlığı	H	1	1	1	2

Figure 7.7 Sample data stored in Term table

7.6.3.4 SynonymTerm Table

Information about the synonym terms of primary terms is stored under SynonymTerm table. 6 fields are defined within this table:

- **OwnerTermID:** Index value of the parent primary term of the synonym term.

- Text: Complete textual representation of the synonym term.
- OwnerTypeID: Index value of the container NE type of the synonym term.
- CategoryID: Index value of the container category of the synonym term.
- IsActive: A bit field which indicates whether the synonym term is enabled for lookup search operations or not.
- NgramCount: Indicates total number of n-grams of the synonym term.

Figure 7.8 shows sample data stored in this table.

OwnerTermID	Text	OwnerTypeID	CategoryID	IsActive	NgramCount
1339	Makedonya Cumhuriyeti	4	65	1	2
1340	Malta Adası	4	65	1	2
1346	Burma	4	72	1	1
1346	Myanmar	4	72	1	1
1356	Büyük Bulgar Devleti	4	89	1	3
1357	Çağatay Devleti	4	89	1	2
1365	Sibir Hanlığı	4	89	1	2
1367	Batı Roma	4	90	1	2
1371	Pontus Krallığı	4	90	1	2
1373	Asya Krallığı	4	90	1	2
1373	Selevkoslar Krallığı	4	90	1	2
1374	Sahip Ataoğulları Beyliği	4	91	1	3
1376	Burhaneddin Ahmet Devleti	4	91	1	3
1376	Burhaneddin Beyliği	4	91	1	2

Figure 7.8 Sample data stored in SynonymTerm table

7.7 Experimental Results

Success of the NER model is tested on the dataset with 60 documents (30 history and 30 geography), which is used on SBD module experiments. Precision and recall metrics for TEXT (to correctly detect borders of NE) and TYPE (to correctly detect type of NE) attributes are used for evaluation. Experiments on geography and history domains are separated to allow comparisons, conclusive results are calculated by combining these two experiment sets. Detected NE types are also counted among correctly guessed type values to compare distributions between different domains.

Precision values are calculated by dividing number of correct guesses to number of all detections, recall values are calculated by dividing number of correct guesses to number of actual named entities. Evaluation metrics used on experiments are formulated below on equations 7.1, 7.2, 7.3 and 7.4.

$$\text{Precision TEXT (\%)} = \frac{100 (\# \text{ of Correct TEXT})}{\# \text{ of Detected NE}} \quad (7.1)$$

$$\text{Precision TYPE (\%)} = \frac{100 (\# \text{ of Correct TYPE})}{\# \text{ of Detected NE}} \quad (7.2)$$

$$\text{Recall TEXT (\%)} = \frac{100 (\# \text{ of Correct TEXT})}{\# \text{ of Actual NE}} \quad (7.3)$$

$$\text{Recall TYPE (\%)} = \frac{100 (\# \text{ of Correct TYPE})}{\# \text{ of Actual NE}} \quad (7.4)$$

Actual number of named entities are determined before performing the experiments. 30 history documents contain 1654, 30 geography documents contain 991 named entities, which makes a grand total of 2645 named entities on 60 documents. Average number of named entities per document is calculated as 55.13 for history domain, 33.03 for geography domain and 44.98 for the combined dataset.

NE type distribution on the test documents are also determined before experimentation. On 30 history documents, there exist 133 Person Turkish, 48 Person Foreign, 273 Location State – Country, 126 Location Other, 101 Organization, 9 Historic Term Building, 127 Historic Term Event, 39 Geographic Term Formation, 221 Date, 26 Date or Number, 5 Percentage and 546 Other tagged named entities. It is observed that no NE with Geographic Term Event tag exists on these documents.

On 30 geography documents, there exist 8 Person Foreign, 225 Location State Country, 200 Location Other, 4 Organization, 3 Historic Term Building, 3 Historic Term Event, 209 Geographic Term Formation, 27 Geographic Term Event, 47 Date, 62 Date or Number, 20 Percentage and 183 Other tagged named entities. It is observed that no NE with Person Name Turkish exists on these documents.

Experiments on history domain resulted in 96.06% precision for TEXT, 92.67% precision for TYPE, 95.83% recall for TEXT and 92.44% recall for TYPE. Experiments on geography domain resulted in 96.59% precision for TEXT, 93.37% precision for TYPE, 97.07% recall for TEXT and 93.84% recall for TYPE. Combined results are **96.26%** precision for TEXT, **92.93%** precision for TYPE, **96.29%** recall for TEXT and **92.97%** recall for TYPE. Table 7.7 and 7.8 shows the combined results.

Table 7.7 Suggestive numerical values derived from NER model experiments

DOMAIN	# of Actual NE	# of Detected NE	# of Correct TEXT	# of Correct TYPE	# of Missed NE
HISTORY Documents (30)	1654	1650	1585	1529	69
GEOGRAPHY Documents (30)	991	996	962	930	25
TOTAL (60 documents)	2645	2646	2547	2459	94
AVG	44.08	44.10	42.45	40.98	1.57

Table 7.8 Precision and recall values derived from NER model experiments

DOMAIN	Precision TEXT (%)	Precision TYPE (%)	Recall TEXT (%)	Recall TYPE (%)
HISTORY Documents (30)	96.06	92.67	95.83	92.44
GEOGRAPHY Documents (30)	96.59	93.37	97.07	93.84
TOTAL (60 documents)	96.26	92.93	96.29	92.97

Results show that success rate for geography domain is slightly better than history domain. But the fact that average number of named entities in a history document is way higher than average number of named entities in a geography document (more than 22) should not be avoided. In both domains, accuracy on TEXT resulted to be higher than accuracy on TYPE, for both precision and recall metrics. Main reason for this is, when the boundaries of a NE is not correctly distinguished, predicting the type of this incorrect text turns out to be an unfeasible task. Ambiguous lexicon terms and person names that can also be used as common nouns are two other issues that cause erroneous detections.

An analysis to detect success rate of the model for individual NE types is also made on experiment results. Table 7.9 compares number of correctly detected NEs for each type with the actual number in history and geography domains, also in the combined data set with 60 documents. For each NE type, average numbers of detected and actual named entities in 60 documents are also included. Accuracy (Acc) value for each NE type t , which is formulated on equation 7.5 is used for evaluation.

System success at detection NEs with Percentage, Date, Location State – Country, Historic Term Event and Other types reached highest accuracy values with 100%, 98.88%, 96.79%, 93.85% and 92.87% respectively. Lowest accuracy value among 13 NE types is observed on Geographic Term Event with 88.89% (24 out of 27).

$$Acc_t(\%) = \frac{100 (\# \text{ of Named Entities Correctly Detected as } t)}{\# \text{ of Actual } t \text{ Typed Named Entities}} \quad (7.5)$$

Table 7.9 NER model experiment results for individual NE types

DOCUMENTS	Detected / Actual	Person Turkish	Person Foreign	Location State/Country	Location Other	Organization	Historic Term Building
HISTORY Docs (30)	Detected	121	43	259	114	90	8
	Actual	133	48	273	126	101	9
GEOGRAPHY Docs (30)	Detected	0	7	223	185	4	3
	Actual	0	8	225	200	4	3
TOTAL (60 docs)	Detected	121	50	482	299	94	11
	Actual	133	56	498	326	105	12
AVG	Detected	2.02	0.83	8.03	4.98	1.57	0.18
	Actual	2.21	0.93	8.30	5.43	1.75	0.20
Accuracy (%)		90.98	89.28	96.79	91.71	89.52	91.67

DOCUMENTS	Detected / Actual	Historic Term Event	Geo Term Formation	Geo Term Event	Date	Date or Number	Percentage	Other
HISTORY Docs (30)	Detected	119	37	0	218	25	5	502
	Actual	127	39	0	221	26	5	546
GEOGRAPHY Docs (30)	Detected	3	185	24	47	55	20	175
	Actual	3	209	27	47	62	20	183
TOTAL (60 docs)	Detected	122	222	24	265	80	25	677
	Actual	130	248	27	268	88	25	729
AVG	Detected	2.03	3.70	0.40	4.42	1.33	0.42	11.28
	Actual	2.16	4.13	0.45	4.46	1.46	0.42	12.15
Accuracy (%)		93.85	89.52	88.89	98.88	90.91	100	92.87

Distribution of correctly detected NE types for both domains is also shown on Table 7.9. Other, Location State – Country, Date, Person Turkish and Historic Term Event are the five most encountered NE types for history documents. Location State – Country, Location Other, Geographic Term Formation, Other and Date or Number are the five most encountered NE types for geography documents. Absence of any Person Turkish tagged NE in geography domain and absence of any Geographic Term Event tagged NE in history domain are remarkable results. Location State – Country appears to be the most homogeneously distributed NE tag among the complete experiment set.

7.8 Encountered Challenges

Problems and restrictions, mostly in connection with Turkish language or common violations in input documents are encountered during the development process of NER model.

Using a wide Turkish first name lexicon provides a high recall in detecting in person names, but it is possible to lead decreases in precision. This is because of the nature of Turkish, as some of the person name words might also indicate common nouns that are frequently used in lecture notes like “*Savaş* (War), *Barış* (Peace), *Nehir* (River), *Irmak* (River)”. Neighbor token controls mostly avoid erroneous detection when these terms are in the beginning of the sentence. In some conditions, these controls are not single-handedly enough. For example, CONJ_SWC lexicon is also beneficial when the first word of a sentence is a conjunction and followed by a NE.

Some expressions like “*Sultan* (Sultan), *Şah* (Shah)” in contextual model might occur both before or after a person name, in fact it is also possible for two conditions to occur at the same time, for example “*Kanuni Sultan Süleyman*” (Suleiman the Magnificent). System used to detect two different named entities in these situations (as “*Kanuni Sultan*” and “*Sultan Süleyman*”), then this is corrected and detected partial expressions are merged to reach the correct NE.

Heading texts are handled with additional controls, as traditionally all heading words (except conjunctions) starts with a capital, even it doesn’t indicate a proper noun. This caused to limit the usage of Other tag for a NE and raised the significance of apostrophe controls.

Separating a commonly used “Person” NE type into two (as Person Turkish and Person Foreign) seems to cause TYPE mistakes in some occasions (which wouldn’t happened if two types are merged as a single Person type). Especially because some first names used in Turkish like “*Musa, Enver, Zeynel, Süleyman*” are also common in Arab countries. Experiments show the performance drops are acceptable though, as

differentiating Turkish and foreign person names is proven to be a rewarding approach for building glossary of term structure phase.

Absence of required punctuation marks (most frequently apostrophe and comma) and spelling errors on input text documents also has negative impacts on system success. It also decreases the quality of named entities and leads to an increased number of Other tagged named entities. For this reason, applying a spell check operation on the document before submitting it as an input is highly recommended.



CHAPTER EIGHT

AUTOMATIZED QUESTION AND TEST GENERATION

8.1 Overview

After the sub-modules, each with an NLP task (document classification, sentence boundary (SBD) and heading detection, verb polarity detection and conversion, named entity recognition (NER) to build a glossary of terms structure) are developed and proven to yield satisfactory results, they are combined to form a single DLL named ITESTCore and served for the examination module usage. This final software, which provides the expected automatized question and test generation functionalities, is named as iTest. This chapter gives detailed information about the contributions of NLP models on question generation process, how to decide between candidate question types, the features of educational software iTest and the infrastructure of this project.

8.2 Automatized Question Generation

8.2.1 Contribution of SBD Model

Every question generation operation is based on a sentence within an input document, which is provided by SBD model. This also gives the examination model idea about the possible number of questions within the constraints specified by criteria. If no constraints are specified, number of questions within a generated exam is the number of sentences derived by SBD model from the input document.

SBD model also provides information about the sub-headings of sentences. This contribution is essential as question text might be a bit vague on its own in some cases. Supporting it with a sub-heading mostly resolves this problem and increases the question quality. On the other hand, in some cases a sub-heading might be more explanatory than expected and implies the correct answer. For this reason, sub-heading feature is made optional and users are allowed to toggle sub-heading visibility any time on examination screen.

8.2.2 Contribution of Document Classification Model

Main contribution of the document classification operation is the automatic classification of the input text document as geography or history based on its domain. This allows system to filter glossary of terms structure and apply search operations on relevant terms instead of using the complete lexicon. Besides, classification result for every input document is stored in database to avoid re-classification of recently loaded, analyzed and classified documents.

Rather than question generation, document classification model is also beneficial for existing exam filtering on test selection phase and exam result filtering on test evaluation phase, as every exam is related to a single classified document. In example use cases, a user can select an existing exam to solve among the ones that are classified as history, or only list his/her exam results on geography course.

8.2.3 Using Verb Polarity for Question Generation

Polarity of the predicate of a sentence directly states the polarity information of that sentence. Correctly classifying a sentence as affirmative or negative, then converting it to the opposite polarity is one of the two ways to obtain a true - false question on the examination model. Generated true - false question might be formed using either the input sentence itself or the converted sentence with the opposite polarity. Table 8.1 shows the general idea behind this approach.

Table 8.1 General approach for true – false question generation using verb polarity information

Input Sentence Polarity	Converted Sentence Polarity	Generated Question Polarity	Correct Answer of Generated Question
Affirmative	Negative	Affirmative	True
Affirmative	Negative	Negative	False
Negative	Affirmative	Affirmative	False
Negative	Affirmative	Negative	True

Every sentence derived from a document by SBD model are processed by verb polarity detection and conversion model and specified to be classifiable or not. If no meaningful result is found or more than a single candidate result are derived, the sentence is specified to be not-classifiable. These sentences are not included in the process to form true - false questions via verb polarity information. This way, generation of a meaningless question is prevented. Figure 8.1 shows an example system output of a verb polarity detection and conversion operation execution on a complete document. As conversion of sentence with the ID value 30 gave multiple results, it won't be used on true - false question generation based on verb polarity.

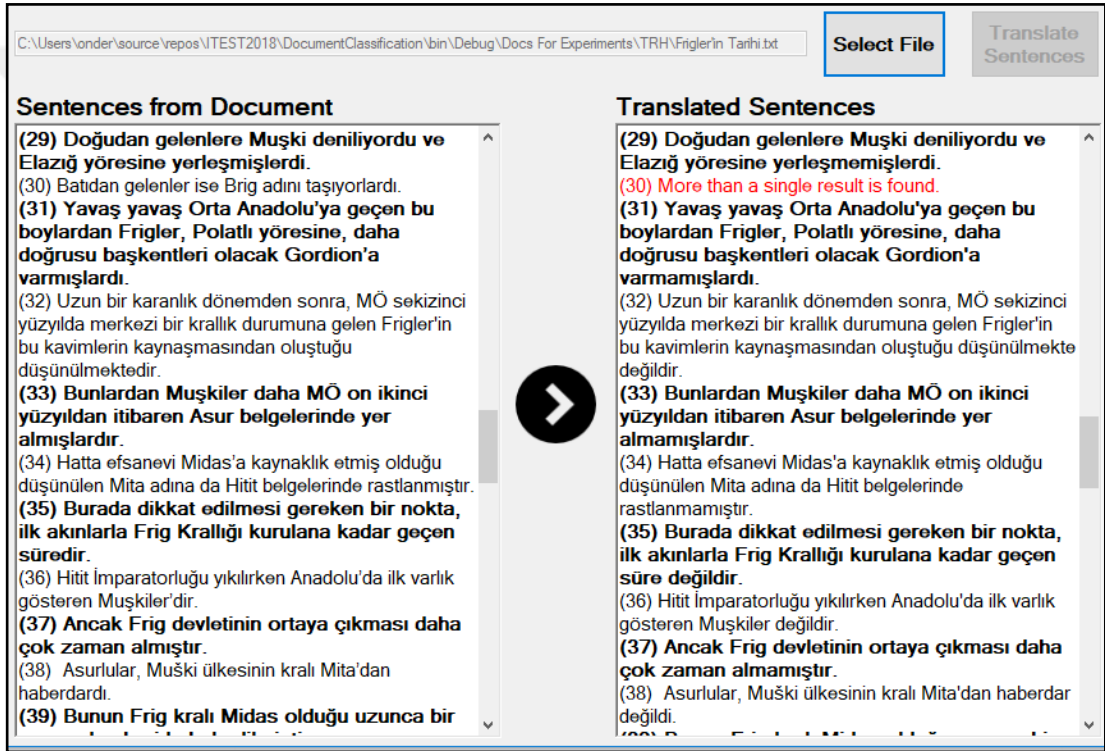


Figure 8.1 Example verb polarity detection and conversion output on a complete document

8.2.4 Using Glossary of Terms for Question Generation

Glossary of terms (GoT) structure, which is formed by NER model execution on 600 history and 600 geography documents is used as a lexicon for lookup operations on examination model. Main approach here is to detect the terms within a sentence that exist in GoT, get the sibling terms of this terms with their total count information

and generate a question. “Sibling term” expression is used to indicate terms with the same fine-grained category and interchangeable with each other in question sentences. Figure 8.2 shows a GoT lookup output on a selected sentence derived from the input document. Two terms are detected within this sentence and the selected one “*Pankuş Meclisi*” (Pankush Council) has 10 sibling terms that are in the scope of the same category named “*Türk ve Dünya Tarihi Meclis*” (Council in Turkish and World History).

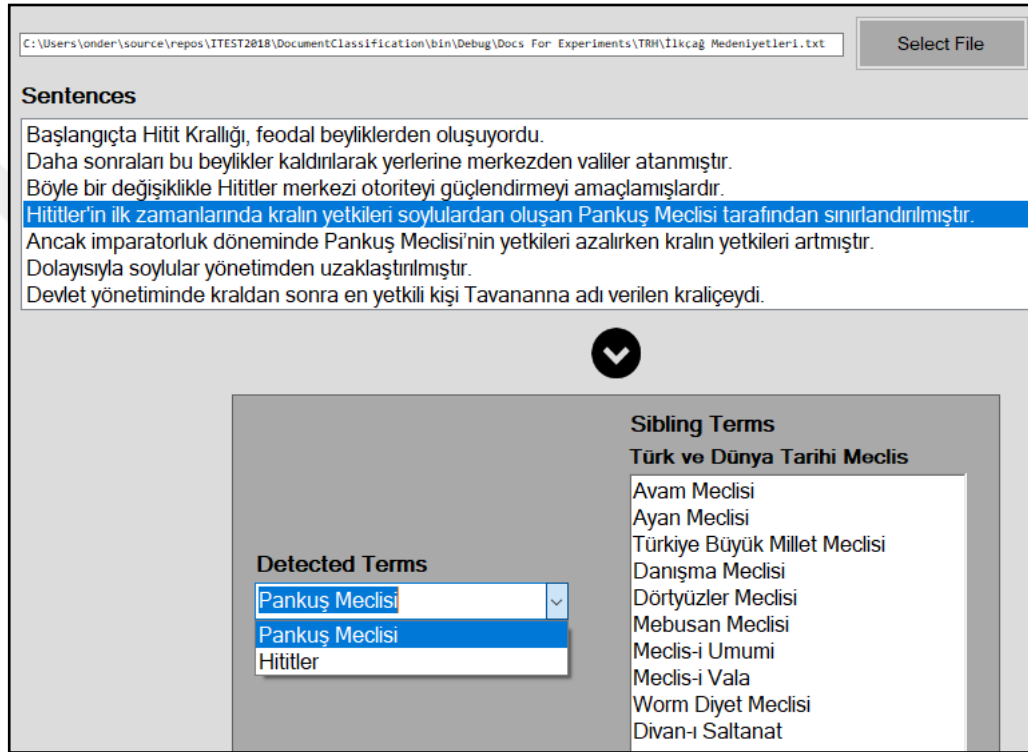


Figure 8.2 Example GoT lookup output to detect terms and their siblings within a sentence

GoT structure is practicable for question generation for all three question types (true - false, fill in the blanks, multiple choice) if the source sentence fits in the required conditions. If at least one term is detected within the source sentence text, that sentence can be considered in generation of a true - false and a fill in the blanks question with GoT, but controls for multiple choice question is still in progress. If a sentence contains at least one term and this term has at least 3 sibling terms, then the conditions for a multiple choice question generation are fulfilled. Figure 8.3 shows three different types of questions generated from the same example sentence shown on Figure 8.2 by GoT lookup operations.

Sentence
Hititler'in ilk zamanlarında kralın yetkileri soylulardan oluşan Pankuş Meclisi tarafından sınırlandırılmıştır.
Generated Questions
True-False Question Hititler'in ilk zamanlarında kralın yetkileri soylulardan oluşan Dörtüzler Meclisi tarafından sınırlandırılmıştır.
Fill in the Blanks Question Hititler'in ilk zamanlarında kralın yetkileri soylulardan oluşan tarafından sınırlandırılmıştır.
Multiple Choice Question Hititler'in ilk zamanlarında kralın yetkileri soylulardan oluşan tarafından sınırlandırılmıştır. A) Dörtüzler Meclisi B) Ayan Meclisi C) Pankuş Meclisi D) Avam Meclisi

Figure 8.3 Three types of questions generated from the same sentence

8.2.5 Question Type Decision

A probabilistic approach is used for question type specification. Five properties, which are collected by verb polarity detection and GoT lookup operations on an input sentence is used as metrics to decide which question type to be selected on question generation based on that sentence:

- **ContainsTerm:** Is set true if at least one term from GoT structure is detected within input sentence.
- **SuitableForMc:** Is set true if at least one detected term has three or more sibling terms.
- **IsClassifiable:** Is set true if verb polarity detection and conversion operation on input sentence yielded a single accurate result.
- **IsPos:** Is set true if input sentence is classified as affirmative.
- **IsNeg:** Is set true if input sentence is classified as negative.

Table 8.2 and Table 8.3 shows the pre-defined question type probabilities based on the values of these five boolean metrics. (IsPos and IsNeg columns indicate the conditions where the value of that property is True.) Probability values are specified for six question types, which are: True – False (Tf) question using verb polarity conversion with answer is set to True, Tf question using verb polarity conversion with answer is set to False, Tf question using a term with answer is set to True, Tf question using a term with answer is set to False, fill in the blanks (Fitb) question using a term as the answer and multiple choice (Mc) question using a term as the answer.

Table 8.2 Specified question type probabilities if value of ContainsTerm property is True

QUESTION TYPE PROBABILITIES	Answer	ContainsTerm-True							
		SuitableForMc-True				SuitableForMc-False			
		IsClassifiable-True		IsClassifiable-False		IsClassifiable-True		IsClassifiable-False	
		IsPos	IsNeg	IsPos	IsNeg	IsPos	IsNeg	IsPos	IsNeg
Tf-UseConversion	True	5%	10%	0%	0%	10%	15%	0%	0%
Tf-UseConversion	False	5%	25%	0%	0%	10%	30%	0%	0%
Tf-UseTerm	True	15%	10%	20%	20%	20%	15%	30%	30%
Tf-UseTerm	False	25%	15%	25%	25%	30%	20%	35%	35%
Fitb	Term	25%	20%	25%	25%	30%	20%	35%	35%
Mc	Term	25%	20%	30%	30%	0%	0%	0%	0%

Table 8.3 Specified question type probabilities if value of ContainsTerm property is False

QUESTION TYPE PROBABILITIES	Answer	ContainsTerm-False			
		SuitableForMc-False			
		IsClassifiable-True		IsClassifiable-False	
		IsPos	IsNeg	IsPos	IsNeg
Tf-UseConversion	True	50%	30%	0%	0%
Tf-UseConversion	False	50%	70%	0%	0%
Tf-UseTerm	True	0%	0%	0%	0%
Tf-UseTerm	False	0%	0%	0%	0%
Fitb	Term	0%	0%	0%	0%
Mc	Term	0%	0%	0%	0%

For example, if ContainsTerm and SuitableForMc properties of an input sentence are set to True and IsClassifiable property is set to False, IsPos and IsNeg properties are not checked. As true - false question generation using verb polarity conversion is not feasible, probability values of two Tf-UseConversion types are set to 0%. Probability values for Tf-UseTerm with answer is set to True is specified as 20%, Tf-

UseTerm with answer is set to False is specified as 25%, Fitb is specified as 25% and Mc is specified as 30%. If user deselects a question type on test preparation phase, probability values of that question type is equally distributed over the included question types. Using the final probability values, question type to be generated for a sentence is randomly determined.

8.3 Test Generation

Test generation is the process where questions that are generated (or selected) considering the specified criteria are combined to obtain a test. Test generation can be done on a new input document or a previously processed document.

8.3.1 Specifiable Criteria

Before starting the test generation process, user is allowed to specify some criteria to adjust the test for his/her needs. Three criteria are defined within the system:

- **Question Types to Include:** By default, all of the possible question types are included for an exam generation, but users are allowed to make changes. For example, if a user wants to generate a test of multiple choice questions only, then he/she can exclude true - false and fill in the blanks question options.
- **Question Limit:** By default, all of the sentences that can be used in question generation are included for an exam generation, but users are allowed to put a limit and specify the desired number of questions. This number should not exceed the maximum number of possible questions, which is determined by the total number of sentences and the included question types. For example, if a user wants a test of multiple choice questions only, and 12 of 25 derived sentences can be used for this purpose, user is not allowed to specify the question limit as 13 or more.
- **Question Order:** By default, order of sentences in their source document is preserved while placing the generated questions into the test. Users are allowed to change it and ignore the sentence order. This can be useful for the conditions where a question text implies the answer of its adjacent question.

8.3.2 Test Generation on New Document

First way to generate a test is to load a text-based course document to the system and specify the criteria based on the user needs. This operation leads to new questions, from which user can select to store for further usage on exam result screen. Document classification of the new document is also done on this stage. New test screen after the classification of input document is completed is shown on Figure 8.4.

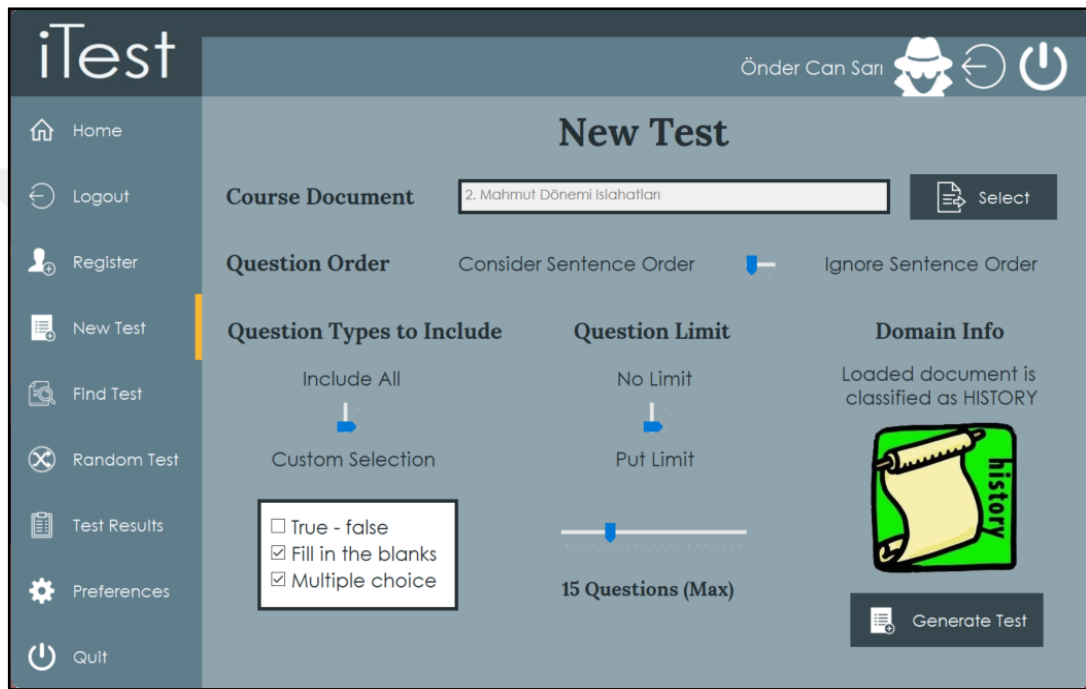


Figure 8.4 Generate test using new input document screen

8.3.3 Test Generation on Existing Document

Second way to generate a test is to use a question pool of a previously loaded and processed text-based course document. Question pool is formed by the previously stored questions by users, after an examination process based on that document. Information about the total number of usable questions for each question type is given to user, so user specifies the criteria considering those constraints. Thus, a proper test is prepared. As this operation uses existing questions and doesn't generate new ones, question save functionality is disabled this time to prevent duplicate questions. Figure 8.5 shows the test generation on an existing document screen.

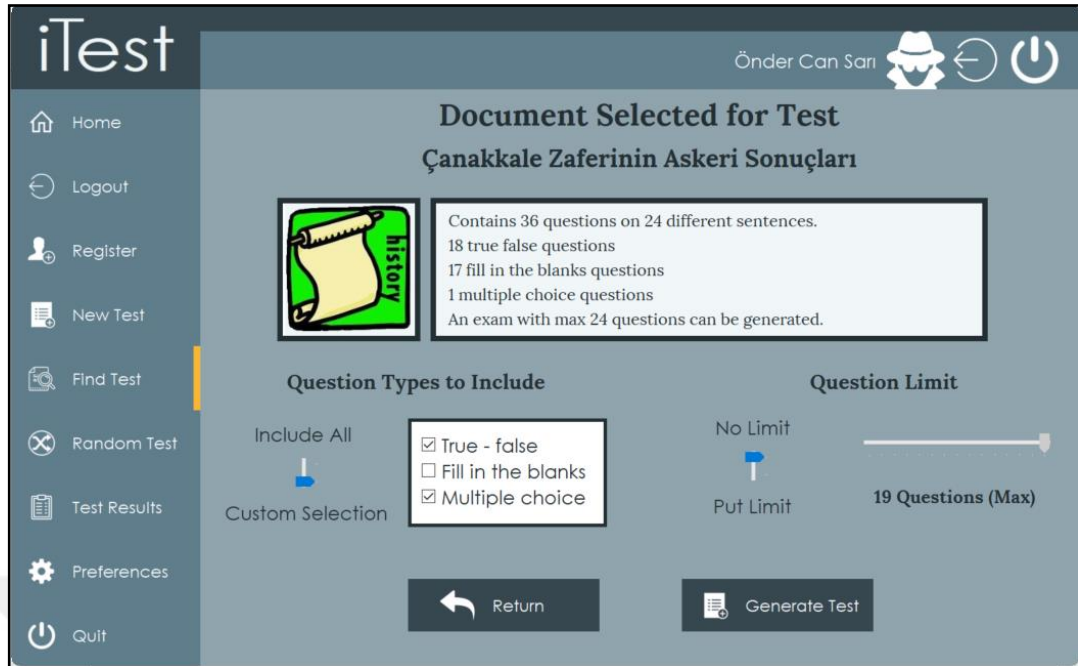


Figure 8.5 Generate test using existing document screen

8.4 Other Features of iTest

Examination model iTest is developed as a complete user-based educational software and many features are provided in this direction.

- **Register - Login Operations:** A user is expected to register and login to the system to benefit from the functionalities of iTest. To complete a registration, user needs to provide a username and a password, to select a profile image between 12 alternatives and check or uncheck the “Show password” and “Remember me” options. Advanced Encryption Standard (AES) is used for encryption and decryption of password. On login screen, users can check the “Keep login info” option to preserve the username and password information for the next login, even after the program is terminated. Figure 8.6 and 8.7 show register and login screens respectively.
- **Functionalities During Examination:** On examination screen, users can navigate through questions using Previous and Next buttons or selecting desired question from the related combobox. There also exists a Next Empty button that allows users to see the nearest unanswered question and a Clear

Answer button to reset the answer of active question. Users can also change the visibility of sub-heading of the active question (if exists) with a trackbar. Active question panel shows the index value of the active question and total number of answered questions. Figure 8.8 shows examination screen. Export as PDF is another functionality within this phase, which is detailed separately.

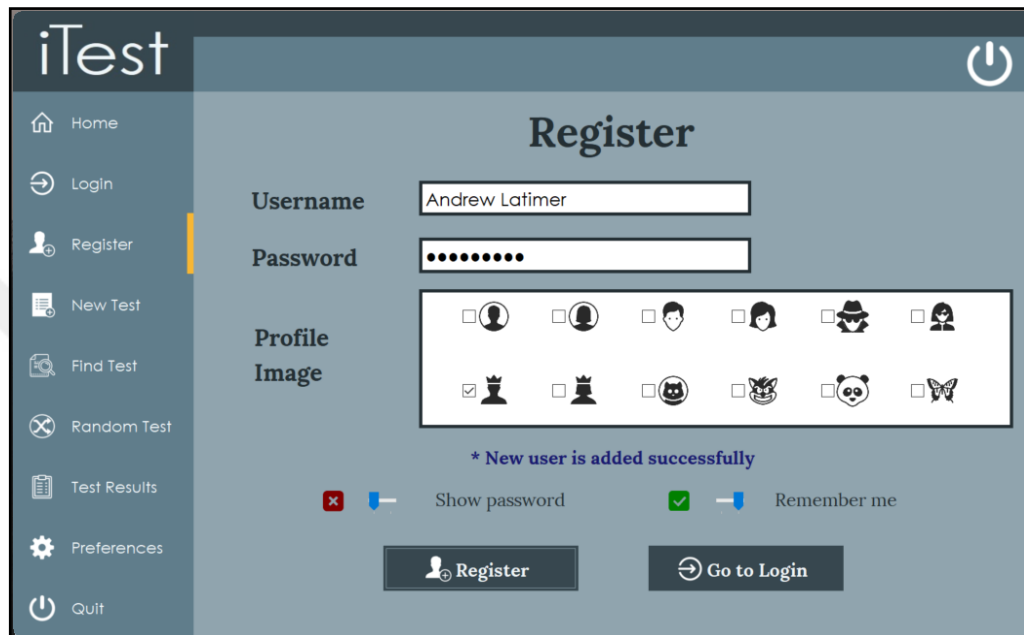


Figure 8.6 Register screen

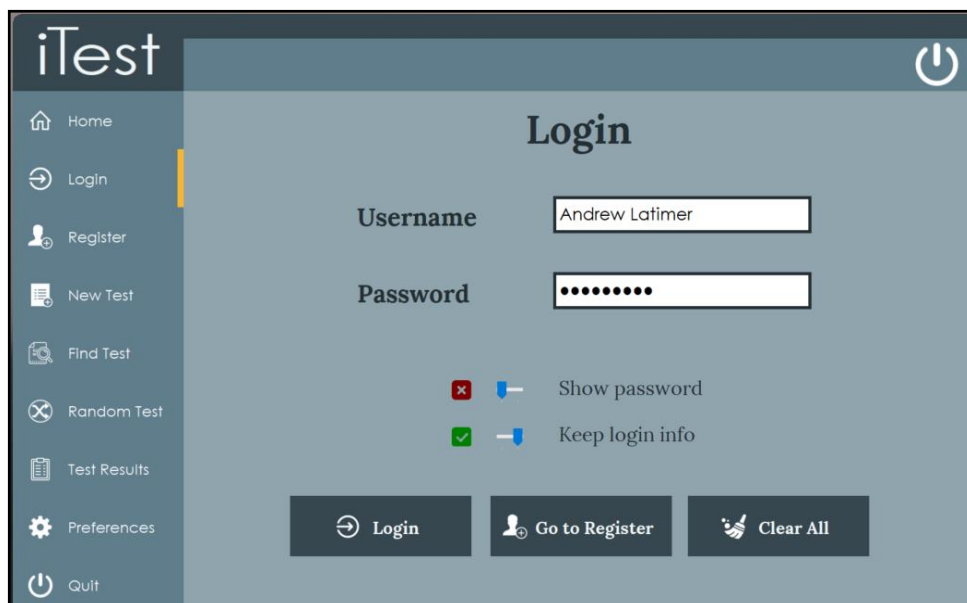


Figure 8.7 Login screen

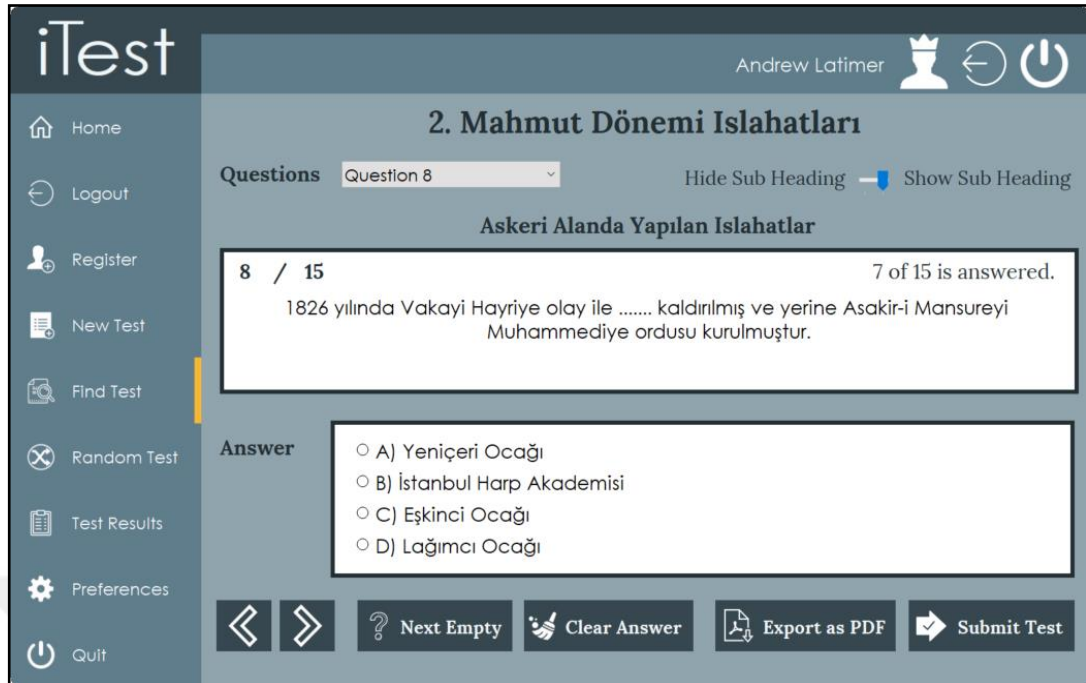


Figure 8.8 Examination screen

- **Export Test as PDF File:** Besides providing a user interface for examination, iTest also allows users to export the generated (or selected) test as a PDF file for a printer friendly review. Pdf files are named by concatenating the related document name, an indicator for the test type (as Mixed, Tf, Fitb, Mc) and the short form of the current date, using a ‘_’ character as a separator. An example text-based course document and the PDF version of one generated test out of that document is given in Appendix-5.
- **Functionalities After Examination:** Exam result screen shows the total number of correct, wrong, unanswered questions and the achieved score. User is allowed to navigate through questions with Previous and Next buttons, also with a combobox. There is also a Next Incorrect button that allows users to see the nearest incorrect (wrong or unanswered) question. Question text, given answer and correct answer of the active question are all displayed on the screen. Users are allowed to save the complete test with all its questions, or select the questions using checkboxes to save for further usage. Note that checkboxes of the previously saved questions are disabled. Saving the exam result is another provided option. Figure 8.9 shows after examination screen.

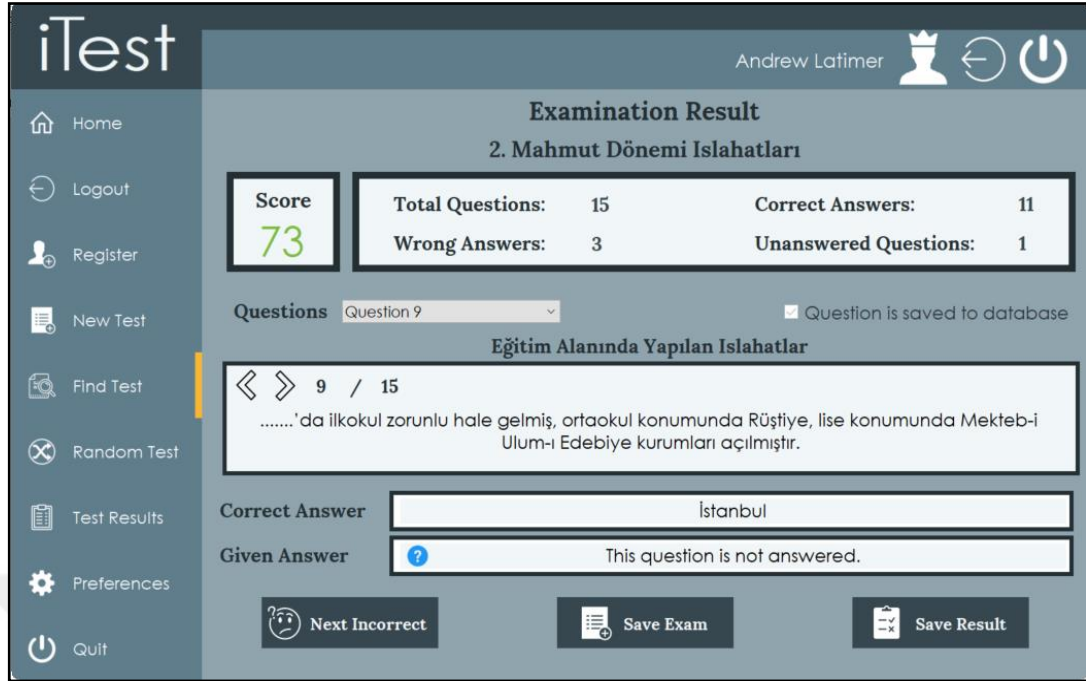


Figure 8.9 After examination screen

- **Find Test Option:** Instead of generating a new exam, users are allowed to choose among the previously stored exams. Available tests can be filtered by domain and test type. Figure 8.10 shows find test screen without filter usage.

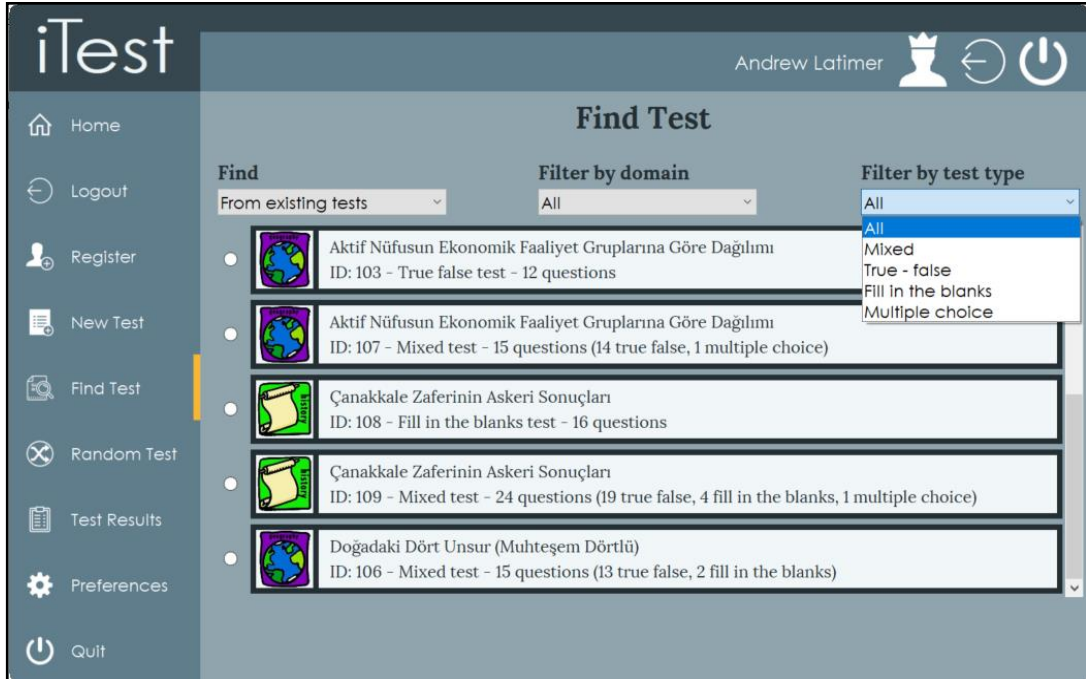


Figure 8.10 Find test screen with no filters are applied

- **Random Test Option:** Another option is to open a random test among the previously stored exams. This feature provides a quick examination opportunity to users without a specification phase.
- **Exam Results Screen:** Users are provided with their saved exam results, which can be filtered by domain and sorted by date (as newest or oldest first) or score (as highest or lowest first). Scores are displayed with different colors depending on the level of success defined by intervals. For example score above 84 are displayed as dark green. Figure 8.11 shows exam results screen when domain filter is applied, and results are sorted by score (as highest first).

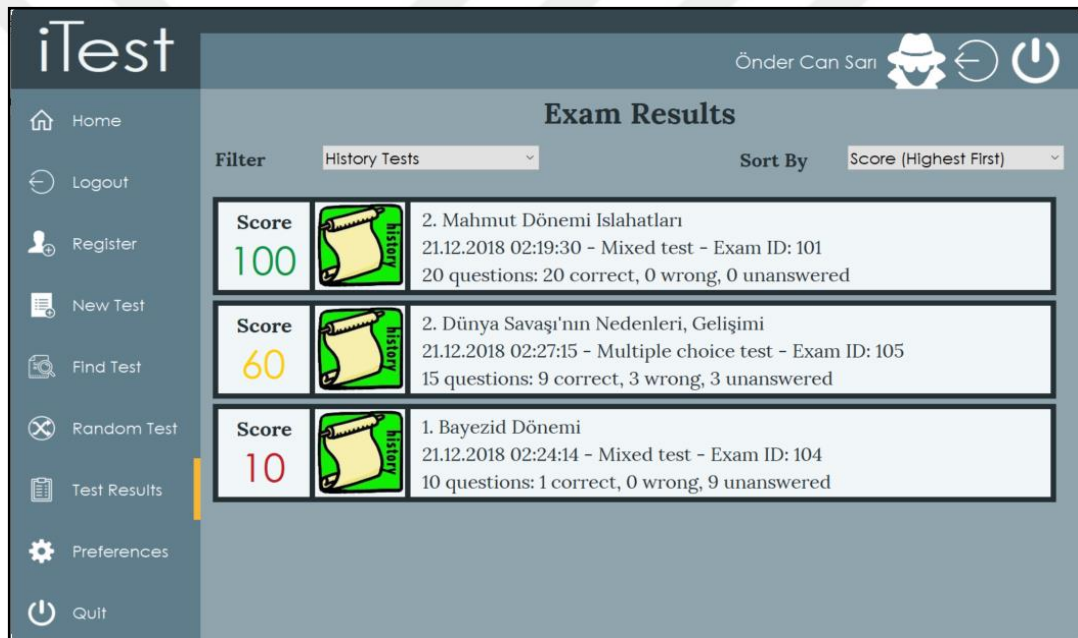


Figure 8.11 Exam results screen when filter and sort operations are applied

- **User-Based Preferences:** Preferences screen allows users to change their profile image, clear their login history and delete their exam results. They can also check or uncheck the “Show redirect screens on page navigation” option, which is checked by default on recently registered users. Redirect screens are used to inform users before page navigation on certain cases. For example, if a user tries to open New Test screen without logging in, system is redirected to Login screen. If the option is enabled, a redirect screen is shown on screen for 2 seconds to inform user before redirect. Figure 8.12 shows preferences screen.

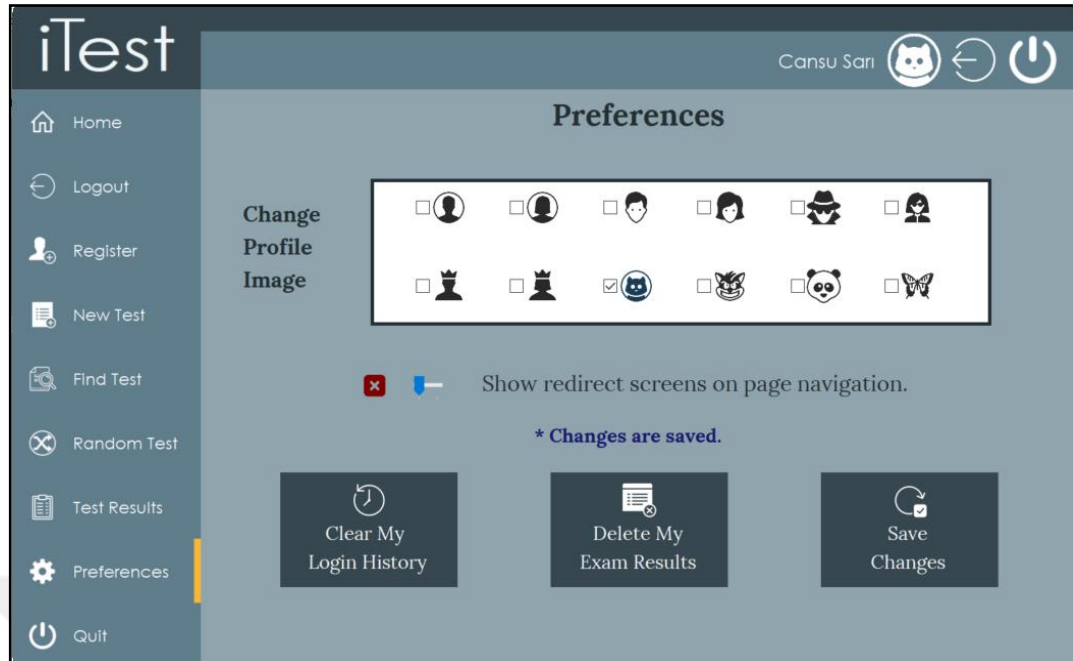


Figure 8.12 Preferences screen

8.5 Database Model

Operations on examination model are highly dependent on a wide range of data, which are stored within the database named ITEST2018. Related data is stored in 8 different tables, which are:

- SystemUser
- UserLogin
- Document
- Exam
- Question
- FitbAnswer
- ExamQuestion
- ExamResult

Database diagram of the model is given in Figure 8.13.

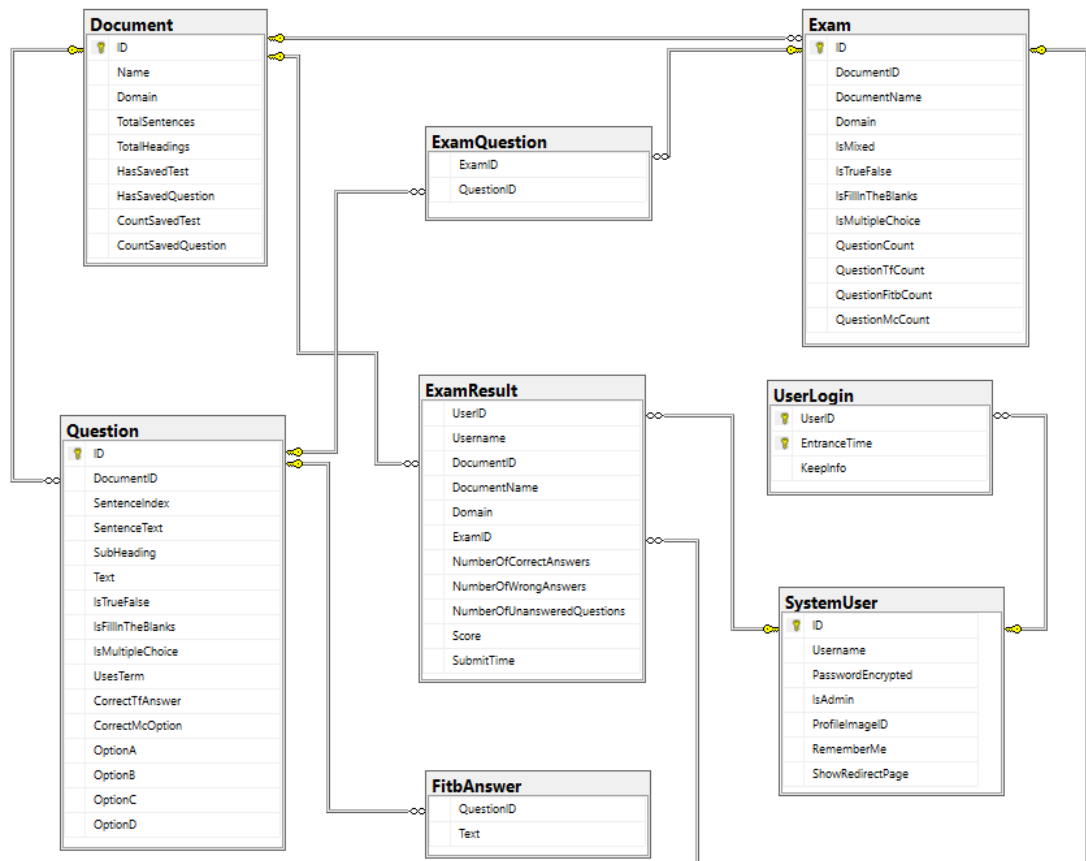


Figure 8.13 Diagram of the ITEST2018 database used for examination model operations

8.5.1 SystemUser Table

Information about the registered users and their preferences is stored under SystemUser table. 7 fields are defined within this table:

- ID: Unique index value of the registered system user.
- Username: Defined username value of the user for system login.
- PasswordEncrypted: Encrypted password value of the user for system login.
- IsAdmin: A bit field indicates whether the system user is an admin or not.
- ProfileImageID: Indicates the index value of the profile image of that user.
- RememberMe: A bit field which indicates whether the user has enabled Remember Me option or not. If enabled, password field is automatically filled after username is correctly typed on login screen.

- ShowRedirectPage: A bit field which indicates whether the user has enabled Show Redirect Page option or not.

8.5.2 UserLogin Table

Login information of registered users is stored under UserLogin table. 3 fields are defined within this table:

- UserID: Index value of the system user that logged in.
- EntranceTime: A datetime field to indicate when the user has logged in.
- KeepInfo: A bit field which indicates whether the user has enabled Keep Info option or not.

8.5.3 Document Table

Information about the documents once given as input and processed by the system are stored under Document table. 9 fields are defined within this table:

- ID: Unique index value of the document entity.
- Name: Name of the input course document file.
- Domain: A nchar(1) field that indicates the domain of the document as 'H' (history) or 'G' (geography).
- TotalSentences: Indicates total number of sentences obtained from that document.
- HasSavedTest: A bit field that indicates whether a complete test, which is generated after processing this particular document, is stored in database or not.
- HasSavedQuestion: A bit field that indicates whether any generated question after processing this particular document is stored in database or not.
- CountSavedTest: Indicates total number of saved tests related with this document.
- CountSavedQuestion: Indicates total number of saved questions related with this document.

8.5.4 Exam Table

Information about the generated exams are stored under Exam table. 12 fields are defined within this table:

- ID: Unique index value of the exam entity.
- DocumentID: Index value of the document which is used to generate this exam.
- Domain: A nchar(1) field that indicates the domain of the exam as 'H' or 'G'.
- IsMixed: A bit field that indicates whether the exam is mixed-type, as it contains more than a single type of questions.
- IsTrueFalse: A bit field that indicates whether the exam contains one or more true false questions.
- IsFillInTheBlanks: A bit field that indicates whether the exam contains one or more fill in the blank questions.
- IsMultipleChoice: A bit field that indicates whether the exam contains one or more multiple choice questions.
- QuestionCount: Indicates total number of questions the exam contains.
- QuestionTfCount: Indicates total number of true false questions the exam contains.
- QuestionFitbCount: Indicates total number of fill in the blanks questions the exam contains.
- QuestionMcCount: Indicates total number of multiple choice questions the exam contains.

Figure 8.14 shows sample data stored in this table.

ID	DocumentID	DocumentName	Domain	IsMixed	IsTrueFalse	IsFillInT...	IsMultipleC...	QuestionCount	QuestionTf...	QuestionFitb...	QuestionMc...
101	100	2. Mahmut Dönemi Islahatları	H	1	1	1	1	20	13	1	6
102	100	2. Mahmut Dönemi Islahatları	H	0	0	1	0	10	0	10	0
103	101	Aktif Nüfusun Ekonomik Faaliyet Grupları...	G	0	1	0	0	12	12	0	0
104	102	1. Bayezid Dönemi	H	1	0	1	1	10	0	7	3
105	103	2. Dünya Savaşı'nın Nedenleri, Gelişimi	H	0	0	0	1	15	0	0	15
106	106	Doğadaki Dört Unsurlar (Muhtesem Dörtlülük)	G	1	1	1	0	15	13	2	0
107	101	Aktif Nüfusun Ekonomik Faaliyet Grupları...	G	1	1	0	1	15	14	0	1
108	107	Çanakkale Zaferinin Askeri Sonuçları	H	0	0	1	0	16	0	16	0
109	107	Çanakkale Zaferinin Askeri Sonuçları	H	1	1	1	1	24	19	4	1

Figure 8.14 Sample data stored in Exam table

8.5.5 Question Table

Information about the generated questions are stored under Question table. 16 fields are defined within this table:

- ID: Unique index value of the question entity.
- DocumentID: Index value of the document which is used to generate this question.
- SentenceIndex: Indicates the index value of the used sentence which implies its order within the owner document.
- SentenceText: Used sentence text which is used to generate question.
- SubHeading: Sub-heading value of the used sentence if one is assigned. NULL is the default value of this field if a sub-heading is not assigned to the sentence.
- Text: Complete textual representation of the question.
- IsTrueFalse: A bit field that is set to 1 if type of the question is true - false and 0 if not.
- IsFillInTheBlanks: A bit field that is set to 1 if type of the question is fill in the blanks and 0 if not.
- IsMultipleChoice: A bit field that is set to 1 if type of the question is multiple choice and 0 if not.
- UsesTerm: A bit field that is set to 1 if the question is generated using a term from GoT and 0 if it is generated by verb polarity information.
- CorrectTfAnswer: A bit field that indicates the correct answer if type of the question is true - false, where 0 implies False and 1 implies True. NULL is the default value if question type is different.
- CorrectMcOption: A nchar(1) field that indicates the correct answer as A, B, C or D, if type of the question is multiple choice. NULL is the default value if question type is different.
- OptionA: Text of Option A, if type of the question is multiple choice. NULL is the default value for this and following 3 fields if question type is different.
- OptionB: Text of Option B, if type of the question is multiple choice.
- OptionC: Text of Option C, if type of the question is multiple choice.
- OptionD: Text of Option D, if type of the question is multiple choice.

8.5.6 FitbAnswer Table

As mentioned before, there exists a table named *SynonymTerm* in *GlossaryOfTerms* database that holds synonyms of primary terms. Every synonym term represents a different variation that can be used instead of a primary term, so multiple correct answers for a fill in the blanks question may exist. Acceptable answers of a fill in the blanks question are stored under *FitbAnswer* table. 2 fields are defined within this table:

- QuestionID: Index value of the parent question entity.
- Text: Complete textual representation of one possible answer.

Figure 8.15 shows how *FitbAnswer* table is used to correctly evaluate the given answer of a fill in the blanks question. As “*Alemdar Mustafa Paşa*” and “*Alemdar Mustafa*” expressions both indicate the same person (an Ottoman grand vizier), system tends to accept any of them as the correct answer. Maximum number of synonym terms of a primary term is determined to be 4. For example, “*Bizans İmparatorluğu*” (Byzantine Empire) can be stated as “*Bizans Devleti*” (Byzantine State), “*Bizans*”, “*Doğu Roma İmparatorluğu*” (East Roman Empire) and “*Doğu Roma*”.

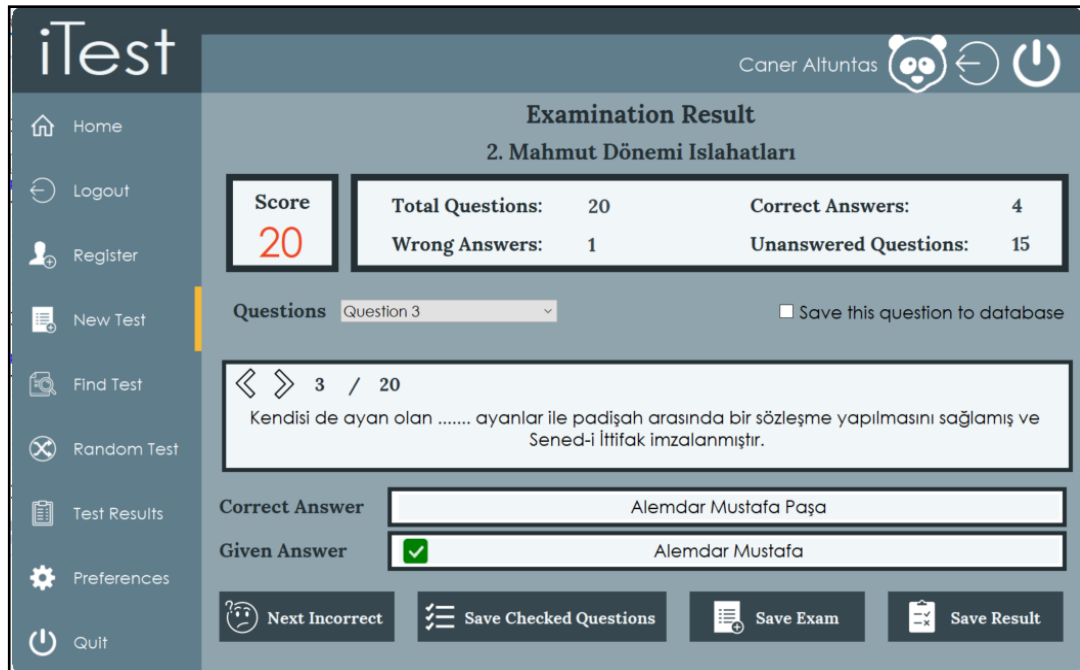


Figure 8.15 Example case where a synonym term is accepted as the correct answer

8.5.7 ExamQuestion Table

As mentioned before, users can either save an exam with all its questions or only save the selected questions with relating them to the source document. Questions of a saved exam entity are stored under ExamQuestion table. 2 fields are defined within this table:

- ExamID: Index value of the related exam entity.
- QuestionID: Index value of the related question entity.

8.5.8 ExamResult Table

Information about exam results are stored under ExamResult table. 11 fields are defined within this table:

- UserID: Index value of the related user entity.
- Username: Username value of the related user entity.
- DocumentID: Index value of the related document entity of which the exam is generated.
- DocumentName: Name value of the related document.
- Domain: A nchar(1) field that indicates the domain of the exam as 'H' or 'G'.
- ExamID: Index value of the related exam entity.
- NumberOfCorrectAnswers: Indicates total number of correct answers user reached on the exam.
- NumberOfWrongAnswers: Indicates total number of wrong answers user did on the exam.
- NumberOfUnansweredQuestions: Indicates total number of questions that user did not give an answer on the exam.
- Score: Indicates the final evaluated score that user reached on the exam.
- SubmitTime: A datetime field to indicate when the user has submitted the exam for evaluation.

CHAPTER NINE

CONCLUSION

9.1 Results and Evaluation

Within this thesis, a computer based examination platform where text-based lecture notes provided by users are analyzed to derive reasonable and meaningful questions and generate an exam is developed. To carry out this goal, history and geography are the included domains for input lecture notes, while true - false, fill in the blanks and multiple choice are the included question types. Besides, users are allowed to specify question types to include, put a limit for number of questions and decide to preserve order of sentences or not while placing questions, before a test generation. Along with the provided opportunities to take an exam, users are also allowed to keep track of their progress by saving their exam results.

Motivation behind the research is the possibility of simple but effective gains like easier access, durability and considerable save of time. The choice of lecture notes is in the user's hands, so possible changes on syllabus won't affect the validity of the project in a negative way. As the generation and storage of the test questions are done in the electronic environment, it offers a paperless self-education opportunity for the students. It is also maintainable as most of the underlying NLP structures are easy to modify.

Besides the examination model, this research draws conclusions and proposes solutions on some of the major NLP tasks for Turkish. Four models each with a different NLP task are developed, tested and finally combined to form the infrastructure of the educational software iTest. Course document classification model is used to automatically detect the domain of the input lecture notes and apply filters based on domain information when needed. Sentence boundary and heading detection model is used to derive sentences and headings, also to join itemized text parts within the input document text. Verb polarity detection and conversion model is one tool to generate true - false questions, in which sentences are classified as affirmative or

negative, then converted to the opposite polarity. Specialized named entity recognition (NER) model is executed on real text-based lecture notes to construct a glossary of terms (GoT) structure in the scope of history and geography domains. GoT structure is a tool to generate questions in all of the three question types.

To perform document classification experiments, a dataset with 1200 text-based course documents (600 geography, 600 history) is collected. In a total of 560 experiments, proportion of data used for training, effect of stop word removal, different stemming approaches, classification algorithms and feature selection methods are compared to select the most suitable model. Based on observations, combination of using 50% of the dataset for training, Naïve Bayes Multinomial (NB-M) as classification algorithm, Zemberek stemmer (ZS) as stemming approach, Information Gain (IG) as feature selection method and removal of stop words (NSW) as the stop word existence approach is chosen as the classification model with its **99.2%** f-measure result.

Combined results for sentence boundary and heading detection model experiments on 60 documents (30 geography, 30 history) are stated as **98.79%** precision for sentences, **98.29%** recall for sentences, **99.35%** precision for headings and **98.50%** recall for headings. All of the 27 cases where a join operation is needed on itemized text parts is are correctly detected and 148 of 158 first character upper - lowercase adjustments are correctly handled.

Combined results for verb polarity detection and conversion model experiments on 60 documents (30 geography, 30 history) are stated as **94.01%** classification accuracy and **92.48%** conversion accuracy.

Success of NER model is evaluated on TEXT (to correctly detect borders of a named entity) and TYPE (to correctly detect type of a named entity) attributes. Combined results for NER model experiments on 60 documents (30 geography, 30 history) are stated as **96.26%** precision for TEXT, **92.93%** precision for TYPE, **96.29%** recall for TEXT and **92.97%** recall for TYPE.

9.2 Future Enhancement

As spelling errors and absence of punctuation marks within documents mostly effect the success rates of the NLP tasks in a negative way, integrating a spell-checker module to the system can be considered before processing the input document.

Verb polarity detection task is handled with an FSM structure that aims to eliminate the inflectional suffixes of a predicate to reach the stem, and a lexicon that holds a wide list of Turkish stems for lookup operations. Even so, because of the agglutinative structure of Turkish language, it is not a realistic approach to store all possible stems in a lexicon structure. Applying derivational suffix controls on the detected stem with a second FSM structure can be considered to reach the root this time, as system success might be increased if lookup operations are executed on a Turkish roots lexicon, which is a more stable list.

On NER model, decreasing the number of named entities with “Other” tag should be considered by additional named entity types. For example, a larger portion of these kind of named entities in history documents have a “nation, nationality” meaning, which can be encapsulated with different tag usage. Lexicons can also be extended with ancient age location and person names.

Working on the detection of word phrases in Turkish sentences, which is another NLP related task, can be considered to increase the number of ways to generate questions on examination model.

REFERENCES

- Aberdeen, J., Burger, J., Day, D., Hirschman, L., Robinson, P., & Vilain, M. (1995). MITRE: Description of the Alembic system used for MUC-6. In *Proceedings of the 6th Conference on Message Understanding*, 141-155. Stroudsburg, PA, USA: ACL.
- Akın, A. A., & Akın, M. D. (2007). Zemberek, an open source NLP framework for Turkic languages. *Structure*, 10, 1-5.
- Aksoy, A., & Öztürk, T. (2016). *TrStop*. Retrieved January 18, 2017, from <https://github.com/ahmetax/trstop/blob/master/dosyalar/turkce-stop-words>.
- Aktaş, Ö. (2010). *Rule-based natural language processing methods for Turkish*. Ph.D. Thesis, Dokuz Eylül University, İzmir.
- Aktaş, Ö., & Çebi, Y. (2013). Rule-based sentence detection method (RBSDM) for Turkish. *International Journal of Language and Linguistics*, 1 (1), 1-6.
- Alfonseca, E., & Manandhar., S. (2002). An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st International Conference on General WordNet*. Mysore, India: GWA.
- Amasyalı, M. F., & Beken, A. (2009). A measurement of Turkish word semantic similarity and text categorization application. In *Proceedings of IEEE 17th Signal Processing and Communications Applications Conference*, 1-4. Antalya, Turkey: IEEE.
- Amasyalı, M. F., & Diri, B. (2006). Automatic Turkish text categorization in terms of author, genre and gender. In C. Kop, G. Fliedl, H. C. Mayr, E. Métais (Eds.), *Proceedings of the 11th International Conference on Applications of Natural Language to Information Systems*, 221-226. Berlin, Heidelberg: Springer-Verlag.

- Baklavas, G., Economides, A. A., & Roumeliotis, M. (1999). Evaluation and comparison of web-based testing tools. In *Proceedings of WebNet-99, World Conference on WWW and Internet*, 81-86. Honolulu, Hawaii: Association for the Advancement of Computing in Education (AACE).
- Bektaş, Y., & Özel, S. A. (2018). The effect of pos tag information on sentence boundary detection in Turkish texts. In *Proceedings of Innovations in Intelligent Systems and Applications Conference (ASYU)*, 1-5. Adana, Turkey: IEEE.
- Cebiroğlu, G. (2002). *Root reaching method without dictionary*. Master's Thesis, İstanbul Technical University, İstanbul.
- Cucerzan, S., & Yarowsky, D. (1999). Language independent named entity recognition combining morphological and contextual evidence. In P. Fung, J. Zhou (Eds.), *Proceedings of Joint SIGDAD Conference on Empirical Methods in NLP and Very Large Corpora*, 90-99. New Brunswick, NJ, USA: ACL.
- Çilden, E. K. (2006). *Stemming Turkish words using Snowball*. Retrieved December 29, 2018, from <http://snowball.tartarus.org/algorithms/turkish/stemmer.html>.
- Çoban, Ö., Özyer, B., & Özyer, G. T. (2015). Sentiment analysis for Turkish Twitter feeds. In *Proceedings of IEEE 23rd Signal Processing and Communications Applications Conference*, 2388-2391. Malatya, Turkey: IEEE.
- Dehkharghani, R., Saygin, Y., Yanikoglu, B., & Oflazer, K. (2016). SentiTurkNet: A Turkish polarity lexicon for sentiment analysis. *Language Resources and Evaluation*, 50, 667-685.
- DePiero, F. (2001). NetExam: A web-based assessment tool for ABET2000. In *Proceedings of 31st Annual Frontiers in Education Conference*, 2, F3A-F13. Reno, NV, USA: IEEE.

- Dinçer, B. T., & Karaoğlu, B. (2003). Stemming in agglutinative languages: A probabilistic stemmer for Turkish. In A. Yazıcı, C. Şener (Eds.), *Proceedings of International Symposium on Computer and Information Sciences (ISCIS 2003), Similarity for conceptual querying (LNCS 2869)*, 244-251. Berlin, Heidelberg: Springer-Verlag.
- Dinçer, B. T., & Karaoğlu, B. (2004). Sentence boundary detection in Turkish. In T. Yakhno (Ed.), *Proceedings of the Third International Conference on Advances in Information Systems (ADVIS '04)*, 255-262. Berlin, Heidelberg: Springer-Verlag.
- Fürnkranz, J. (1998). *A study using n-gram features for text categorization* (Report No. OEFAI-TR-98-30). Wien, Austria: Austrian Research Institute for Artificial Intelligence.
- Gotoh, Y., & Renals, S. (2000). Sentence boundary detection in broadcast speech transcripts. In *Proceedings of ISCA Workshop: Automatic Speech Recognition: Challenges for the New Millennium (ASR-2000)*, 228-235. Paris, France: International Speech Communication Association (ISCA).
- Grishman, R., & Sundheim, B. (1996). Message Understanding Conference-6: A brief history. In *Proceedings of the 16th Conference of Computational Linguistics (COLING '96)*, 1, 466-471. Stroudsburg, PA, USA: ACL.
- Grün, B., & Zeileis, A. (2009). Automatic generation of exams in R. *Journal of Statistical Software*, 29 (10), 1-14.
- Guz, U., Favre, B., Hakkani-Tur, D., & Tur, G. (2009). Generative and discriminative methods using morphological information for sentence segmentation of Turkish. *IEEE Transactions on Audio, Speech, and Language Processing*, 17, 895-903.

- Güneş, A., Tantuğ, A. C. (2018). Turkish named entity recognition with deep learning. In *26th Signal Processing and Communications Applications Conference (SIU)*, 1-4. İzmir, Turkey: IEEE.
- Güngör, O., Üsküdarlı, S., & Güngör, T. (2018). Recurrent neural networks for Turkish named entity recognition. In *26th Signal Processing and Communications Applications Conference (SIU)*, 1-4. İzmir, Turkey: IEEE.
- Han, E., & Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In D. Zighed, H. Komorowski, J. Zytkow (Eds.), *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, 424-431. London, UK: Springer-Verlag.
- Hussein, H., Elmogy, M., & Guirguis, S. (2014). Automatic English question generation system based on template driven scheme. *International Journal of Computer Science Issues (IJCSI)*, 11 (6), 45-53.
- Isa, D., Lee, L. H., Kallimani, V. P., & RajKumar, R. (2008). Text document preprocessing with the Bayes formula for classification using the support vector machine. *IEEE Transaction on Knowledge and Data Engineering*, 20, 1264-1272.
- Jurafsky, D., & Martin, J. H. (2000). Morphology and finite-state transducers. In *Speech and language processing* (57-90). Englewood Cliffs, New Jersey: Prentice-Hall, Inc.
- Jurafsky, D., & Martin, J. H. (2009). Named entity recognition. In *Speech and language processing* (2nd ed.) (743-751). Upper Saddle River, New Jersey: Prentice-Hall, Inc.
- Kesgin, F. (2007). *Türkçe metinler için konu belirleme sistemi*. Master's Thesis, İstanbul Technical University, İstanbul.

- Kiss, T., & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32, 485-525.
- Küçük, D., & Yazıcı, A. (2009). Named entity recognition experiments on Turkish texts. In T. Andreasen, R. R. Yager, H. Bulskov, H. Christiansen, H. L. Larsen (Eds.), *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, 524-535. Berlin, Heidelberg: Springer-Verlag.
- Küçük, D., & Yazıcı, A. (2009). Rule-based named entity recognition from Turkish texts. In *Proceedings of International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2009)*, 456-460. Trabzon, Turkey: Karadeniz Technical University Press.
- Küçük, D., & Yazıcı, A. (2012). A hybrid named entity recognizer for Turkish. *Expert Systems with Applications*, 39, 2733-2742.
- Küçük, D., Jacquet, G., & Steinberger, R. (2014). Named entity recognition on Turkish tweets. In N. Calzolari et al. (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC '14)*, 450-454. Reykjavik, Iceland: ELRA.
- Küçük, D., Küçük, D., & Arıcı, N. (2016). A named entity recognition dataset for Turkish. In *Proceedings of 24th Signal Processing and Communication Application Conference (SIU)*, 329-332. Zonguldak, Turkey: IEEE.
- Lee, J. (2000). *Internet-based exam generator system for review of the fundamentals of engineering exam*. Master's Thesis, University of Oklahoma, Oklahoma.
- Lewis, D. D. (1992). Feature selection and feature extraction for text categorization. In M. P. Marcus (Ed.), *Proceedings of the workshop on Speech and Natural Language (HLT '91)*, 212-217. Harriman, New York: Association for Computational Linguistics (ACL).

- Liu, Y., & Shriberg, E. (2007). Comparing evaluation metrics for sentence boundary detection. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '07)*, IV-185 – IV-188. Honolulu, HI, USA: IEEE.
- Liu, Z., Shi, J., Liu, J., & Li, Y. (2010). Strategy and applied research of multi-constrained model of automatic test paper based on genetic algorithm. *Applied Mechanics and Materials*, 37-38, 1223-1230.
- Manevitz, L. M., & Yousef, M. (2002). One-class SVMs for document classification. *The Journal of Machine Learning Research*, 2 (3/1/2002), 139-154.
- Manning, C. D., Raghavan, P., & Schütze, H. (2009). Determining the vocabulary of terms. In *An introduction to information retrieval* (22-35). Cambridge, England: Cambridge UP.
- Mikheev, A. (2000). Tagging sentence boundaries. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, 264-271. Stroudsburg, PA, USA: ACL.
- Moral, C., de Antonio, A., Imbert, R., & Ramirez, J. (2014). A survey of stemming algorithms in information retrieval. *Information Research*, 19 (1), paper 605. Retrieved December 29, 2018, from <http://www.informationr.net/ir/19-1/paper605.html#.XGF7o1wzYdU>.
- Oflazer, K. (2003). Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29, 515-544.
- Oflazer, K. (2014). Turkish and its challenges for language processing. *Language Resources and Evaluation*, 48, 639-653.

- Ozkul, A. (2009). Using information technology to enhance assessment of learning: Automating preparation of course exam materials and student feedback. *Information Technology, Learning and Performance Journal*, 25 (1), 15-23.
- Ozturkmenoglu, O., & Alpkocak, A. (2012). Comparison of different lemmatization approaches for information retrieval on Turkish text collection. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*, 1-5. Trabzon, Turkey: IEEE.
- Özsert, C. M., & Özgür, A. (2013). Word polarity detection using a multilingual approach. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing (CicLing 2013), Part II, Lecture Notes in Computer Science (LNCS 7817)*, 75-82. Berlin, Heidelberg: Springer-Verlag.
- Palmer, D. D., & Hearst, M. A. (1997). Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23, 241-267.
- Porter, M. F. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 40, 130-137.
- Read, J., Dridan, R., Oepen, S., & Solberg, L. J. (2012). Sentence boundary detection: A long solved problem?. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012)*, 985-994. Mumbai, India: ACL.
- Reynar, J. C., & Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 16-19. Stroudsburg, PA, USA: ACL.
- Riley, M. D. (1989). Some applications of tree-based modelling to speech and language. In *Proceedings of the workshop on Speech and Natural Language (HLT '89)*, 339-352. Stroudsburg, PA, USA: ACL.

- Sahin, H. B., Tirkaz, C., Yildiz, E., Eren, M. T., & Sonmez, O. (2017). *Automatically annotated Turkish corpus for named entity recognition and text categorization using large-scale gazetteers*. Retrieved December 26, 2018, from <https://arxiv.org/abs/1702.02363>.
- Sak, H., Güngör, T., & Saraçlar, M. (2008). Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In B. Nordström, A. Ranta (Eds.), *Proceedings of the 6th International Conference on Advances in Natural Language Processing*, 417-427. Berlin, Heidelberg: Springer-Verlag.
- Sang, E., & Meulder, F. (2003). Introduction to CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on NLP at Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, 4, 142-147. Stroudsburg, PA, USA: ACL.
- Sever, H., & Bitirim, Y. (2003). FindStem: Analysis and evaluation of a Turkish stemming algorithm. In M. A. Nascimento, E. S. de Moura, A. L. Oliveira (Eds.), *Proceedings of String Processing and Information Retrieval (SPIRE 2003), Lecture Notes in Computer Science (LNCS 2857)*, 238-251. Berlin, Heidelberg: Springer-Verlag.
- Shende, A. M., Dalch, L. N., & Warner, M. R. (2002). United States Patent No. US 6341212 B1. Retrieved from <https://patents.google.com/patent/US6341212>.
- Slonim, N., Friedman, N., & Tishby, N. (2002). Unsupervised document classification using sequential information maximization. In K. Jarvelin, M. Beaulieu, R. Baeza-Yates, S. Myaeng (Eds.), *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 129-136. New York, NY, USA: ACM.

- Solak, A., & Can, F. (1994). *Effects of stemming on Turkish text retrieval* (Report No. BUCEIS-94-20). Ankara, Turkey: Bilkent University.
- Solak, A., & Oflazer, K. (1993). Design and implementation of a spelling checker for Turkish. *Literary and Linguistic Computing*, 8, 113-130.
- Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (1999). Automatic extraction of rules for sentence boundary disambiguation. In *Proceedings of Workshop on Machine Learning in Human Language Technology, Advance Course in Artificial Intelligence (ACAI '99)*, 88-92. Crete, Greece: Mediterranean Agronomic Institute of Chania.
- Şahin, M., Sulubacak, U., & Eryiğit, G. (2013). Redefinition of Turkish morphology using flag diacritics. In *Proceedings of the 10th Symposium on Natural Language Processing (SNLP-2013)*. Phuket, Thailand: SIIT, NECTEC.
- Şeker, G., & Eryiğit, G. (2016). *State of the art in Turkish named entity recognition*. Retrieved March 9, 2018, from <https://pdfs.semanticscholar.org/7e7f/ed9d21a3e3a36c4eb3c7df1ee8116e8ec2ce.pdf>.
- Şeker, G.A., & Eryiğit, G. (2012). Initial explorations on using CRFs for Turkish named entity recognition. In *Proceedings of 24th International Conference on Computational Linguistics (COLING 2012)*, 2459-2574. Mumbai, India: The COLING 2012 Organizing Committee.
- Tantuğ, A. C. (2010). Document categorization with modified statistical language models for agglutinative languages. *International Journal of Computational Intelligence System*, 3, 632-645.

- Tatar, S., & Cicekli, I. (2011). Automatic rule learning exploiting morphological features for named entity recognition in Turkish. *Journal of Information Science*, 37, 137-151.
- Temizsoy, M., & Cicekli, I. (1998). An ontology-based approach to parsing Turkish sentences. In D. Farwell, L. Gerber, E. Hovy (Eds.), *Proceedings of Antenna Measurement Techniques Association 20th Annual Meeting and Symposium (AMTA '98)*, 124-135. London, UK: Springer-Verlag.
- The most influential people of all time.* (n.d). Retrieved May 12, 2018, from <https://www.ranker.com/crowdranked-list/the-most-influential-people-of-all-time>.
- Ting, S. L., Ip, W. H., & Tsang, A. (2011). Is naïve bayes a good classifier for document classification?. *International Journal of Software Engineering and Its Applications*, 5 (3), 37-46.
- Tüfekçi, P., Uzun, E., & Sevinç, B. (2012). Text classification of web based news articles by using Turkish grammatical features. In *Proceedings of IEEE 20th Signal Processing and Communications Applications Conference*, 1-4. Mugla, Turkey: IEEE.
- Tür, G., Hakkani-Tür, D., & Oflazer, K. (2003). A statistical information extraction system for Turkish. *Natural Language Engineering*, 9, 181-210.
- Ugurdag, H. F., Argali, E., Eker, O. E., Basaran, A., Goren, S., & Özcan, H. (2009). Smart question (sQ): Tool for generating multiple-choice test questions. In R. Revetria, V. Mladenov, N. Mastorakis (Eds.), *Proceedings of the 8th WSEAS International Conference on Education and Educational Technology*, 173-177. Genova, Italy: WSEAS Press.
- Uysal, A. K., & Gunal, S. (2014). Text classification using genetic algorithm oriented latent semantic features. *Expert Systems with Applications*, 41, 5938-5947.

- Uysal, A. K., Günal, S., Ergin, S., & Günal, E. Ş. (2012). Detection of SMS spam messages on mobile phones. In *Proceedings of IEEE 20th Signal Processing and Communications Applications Conference*, 1-4. Mugla, Turkey: IEEE.
- Vural, A. G., Cambazoglu, B. B., & Senkul, P. (2012) A framework for sentiment analysis in Turkish: Application to polarity detection of movie reviews in Turkish. In E. Gelenbe, R. Lent (Eds.), *Computer and Information Sciences III*, 437-445. London, UK: Springer-Verlag.
- Wang, H., & Huang, Y. (2003). *Bondec – A Sentence Boundary Detector*. Retrieved March 13, 2018, from https://nlp.stanford.edu/courses/cs224n/2003/fp/huangy/final_project.doc.
- Wentland, W., Knopp, J., Silberer, C., & Hartung, M. (2008). Building a multilingual lexical resource for named entity disambiguation, translation and transliteration. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, 3230-3237. Marrakech, Morocco: ELRA.
- Xu, C., Xie, L., Huang, G., Xiao, X., Chng, E. S., & Li, H. (2014). A deep neural network approach for sentence boundary detection in broadcast news. In *Proceedings of 15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*, 2887-2891. Singapore: ISCA.
- Xu, J., & Croft, W. B. (1998). Corpus-based stemming using co-occurrence of word variants. *ACM Transactions on Information Systems*, 16 (1), 61-81.
- Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. In K. Knight, A. Nenkova, O. Rambow (Eds.), *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, 1480-1489. San Diego, California: ACL.

Yıldırım, S., & Yıldız, T. (2018). A comparative analysis of text classification for Turkish language. *Pamukkale University Journal of Engineering Sciences*, 24, 879-886.

Yildiz, H. K., Gençtav, M., Usta, N., Diri, B. & Amasyalı, M. F. (2007). A new feature extraction method for text classification. In *Proceedings of IEEE 15th Signal Processing and Communications Applications*, 1-4. Eskisehir, Turkey: IEEE.



APPENDICES

APPENDIX-1: Turkish Stop Words

a	birçoğu	da	etmesi	herkesin
acaba	birçok	daha	etti	hiç
altı	biri	dahi	ettiği	hiçbir
altmış	birisi	dan	ettiğini	hiçbiri
ama	birkaç	de	fakat	i
ancak	birşey	defa	filan	ı
arada	biz	değil	filan	için
artık	bizden	diğer	gene	içinde
asla	bize	diğeri	gereği	iki
aslında	bizi	diğerleri	gerek	ile
ayrıca	bizim	diye	gibi	ilgili
az	böyle	doksan	göre	ise
bana	böylece	dokuz	hala	işte
bazen	bu	dolayı	halde	itibaren
bazı	buna	dolayısıyla	halen	itibariyle
bazıları	bunda	dört	hangi	kaç
belki	bundan	e	hangisi	kadar
ben	bunlar	edecek	hani	karşın
benden	bunları	eden	hatta	kendi
beni	bunların	ederek	hem	kendilerine
benim	bunu	edilecek	henüz	kendine
beri	bunun	ediliyor	hep	kendini
beş	burada	edilmesi	hepsi	kendisi
bile	bütün	ediyor	her	kendisine
bilhassa	çoğu	eğer	herhangi	kendisini
bin	çoğunu	elbette	herkes	kez
bir	çok	elli	herkese	ki
biraz	çünkü	en	herkesi	kim

APPENDIX-1 continues

kimse	olduklarını	oysa	ta	ye
kırk	olmadı	pek	tabi	yedi
madem	olmadığı	rağmen	tam	yerine
mi	olmak	sana	tamam	yetmiş
mı	olması	sanki	tamamen	yi
milyar	olmayan	şayet	tarafından	yı
milyon	olmaz	şekilde	trilyon	yine
mu	olsa	sekiz	tüm	yirmi
mü	olsun	seksen	tümü	yoksa
nasıl	olup	sen	u	yu
ne	olur	senden	ü	yüz
neden	olursa	seni	üç	zaten
nedenle	oluyor	senin	un	zira
nerde	on	şey	ün	
nerede	ön	şeyden	üzere	
nereye	ona	şeye	var	
neyse	önce	şeyi	vardı	
niçin	ondan	şeyler	ve	
nin	onlar	şimdi	veya	
nın	onlara	siz	ya	
niye	onlardan	sizden	yani	
nun	onları	size	yapacak	
nün	onların	sizi	yapılan	
o	onu	sizin	yapılması	
öbür	onun	sonra	yapıyor	
olan	orada	şöyle	yapmak	
olarak	öte	şu	yaptı	
oldu	ötürü	şuna	yaptığı	
olduğu	otuz	şunları	yaptığını	
olduğunu	öyle	şunu	yaptıkları	

APPENDIX-2: Most Distinctive 100 Words for Classification

Rank	Word	IG Score
1	devlet	0.52314
2	savaş	0.3528
3	karşı	0.208
4	fazla	0.20644
5	yağ	0.19711
6	sıcaklık	0.19198
7	ordu	0.18133
8	iklim	0.18096
9	iste	0.17603
10	birlik	0.17435
11	ilk	0.16736
12	bitki	0.1655
13	yönetim	0.16336
14	kur	0.16098
15	antlaşma	0.16006
16	başla	0.15773
17	örtü	0.14888
18	kurul	0.1443
19	barış	0.14343
20	karar	0.14219
21	akarsu	0.14201
22	su	0.14028
23	kabul	0.13999
24	askeri	0.13751
25	işgal	0.13523

Rank	Word	IG Score
26	yüksek	0.12536
27	yeni	0.12512
28	yer	0.12491
29	egemen	0.12348
30	yeryüzü	0.12235
31	halk	0.12004
32	asker	0.11969
33	katıl	0.1186
34	siyasi	0.1186
35	amaç	0.11776
36	imzala	0.11636
37	imparator	0.11458
38	ver	0.11282
39	başarı	0.11099
40	dağ	0.11096
41	son	0.1109
42	bağım	0.11014
43	görev	0.10852
45	ilan	0.10826
44	millet	0.10826
46	kaldır	0.10761
47	başkan	0.10582
48	meclis	0.10511
49	paşa	0.10485
50	mevsim	0.10462

APPENDIX-2 continues

Rank	Word	IG Score
51	el	0.10278
52	yükselti	0.10274
53	kemal	0.09889
54	gönder	0.09751
55	saldırı	0.09697
56	güven	0.09463
57	anlaşma	0.09434
58	kazan	0.09271
59	aç	0.0923
60	rüzgar	0.09149
61	yönet	0.09096
62	orman	0.09096
63	er	0.08781
64	milli	0.08768
65	silah	0.08694
66	kış	0.08634
67	çıktı	0.0856
68	kıyı	0.08509
69	nem	0.08502
70	ele	0.0828
71	ortalama	0.08279
72	yardım	0.08228
73	düzenle	0.08227
74	politika	0.08223
75	kurak	0.08176

Rank	Word	IG Score
76	tanı	0.08162
77	hak	0.08156
78	sıcak	0.08153
79	kurt	0.08131
80	gir	0.07885
81	padişah	0.07846
82	girdi	0.07813
83	komutan	0.07813
84	din	0.07813
85	yamaç	0.07752
86	mücadele	0.07718
87	yun	0.07697
88	dönem	0.07607
89	bey	0.07596
90	oluş	0.07593
91	getir	0.0758
92	kutup	0.07561
93	tarihi	0.07561
94	doğal	0.07561
95	üye	0.0756
96	enlem	0.07536
97	düşünce	0.07518
98	yüzey	0.07475
99	kral	0.07453
100	al	0.07387

APPENDIX-3: Turkish Abbreviation List

age.	Bulg.	fizy.	koor.	Ord.	Tğm.
agm.	Cad.	Fr.	Kor.	Org.	tic.
agy.	cm.	g.	Kora.	Ort.	tiy.
Alb.	coğ.	Gen.	Korg.	Osm. T.	tlks.
Alm.	Cum. Bşk.	geom.	kr.	öl.	tls.
anat.	çev.	gn.	krş.	ör.	Top.
ant.	Çvş.	Gnkur.	Kur. Bşk.	Ör.	Tug.
Apt.	dal.	Gön.	Kur.	öz.	Tuğa.
Ar.	dam.	gr.	l.	ped.	Tuğg.
ark.	db.	hay. b.	lt.	Port.	Tüm.
Arş. Gör.	dg.	haz.	Lat.	Prof.	Tüma.
As. İz.	dil b.	hek.	Ltd.	psikol.	Tümg.
As.	dk.	hl.	m.	Rum.	Uzm.
Asb.	dl.	hlk.	Mac.	Rus.	Üni.
astr.	dm	hm.	Mah.	s.	Ü.
astrol.	Doç.	Hs. Uzm.	man.	sa.	Üçvş.
Atğm.	doğ.	huk.	mat.	Sb.	ünl.
atm.	Dr.	Hv. Kuv. K.	Md.	SEFD Bşk.	Ütğm.
Av.	drl.	Hv. Kuv.	mec.	sf.	vb.
bağ.	Dz. Kuv. K.	Hız. öz.	MHz.	Sl.	vd.
Bçvş.	Dz. Kuv.	Hız.	mim.	Sn.	Vet.
bit. b.	dzl.	İbr.	min.	snt.	vs.
biy.	Ecz.	İng.	mm.	Sok.	Y. Mim.
bk.	ed.	is.	Müh.	sos.	Y. Müh.
bkz.	e.	İsp.	Mür.	sp.	Yay.
bl.	ekon.	işl.	müz.	Srp.	Yb.
Bl.	Ens.	İt.	No.	Şb.	Yd. Sb.
Bn.	Erm.	Jap.	Nö.	T.C.	Yrd. Doç.
Bnb.	f.	jeol.	Nö. Sb.	T.	Yun.
bot.	Fak.	kal.	Okt.	tar.	yy.
Böl.	Far.	kg.	Onb.	Tb.	Yzb.
bs.	fel.	KHz.	Opr.	tek.	zf.
Bşk.	fil.	kim.	Or.	tel.	zm.
Bul.	fiz.	km.	Ora.	telg.	zool.

APPENDIX-4: Exceptional Verb Roots Affected by Consonant Lenition

git	vadet	akdet
et	mahvet	akset
tat	cezbet	atfet
güt	şükret	faslet
dit	bahşet	fethet
affet	defet	gasbet
azlet	katet	hatmet
kaybet	hazmet	hicvet
farket	azmet	hazzet
süregit	vehmet	kahret
emret	vakfet	kastet
bahset	zikret	kaydet
hallet	zehret	keşfet
hapset	zulmet	küfret
hükmet	cebret	lağvet
sabret	celbet	methet
hisset	resmet	meylet
devret	feshet	nakşet
zannet	haczet	raptet
katlet	hamdet	reddet
naklet	ahdet	tabet

APPENDIX-5: Example Lecture Note and Generated Test

FENİKE MEDENİYETİ – İBRANİ MEDENİYETİ

FENİKE MEDENİYETİ

Lübnan Dağları ile Akdeniz sahili arasındaki bölgede yaşamış, gemicilik ve ticaretle gelişmiş bir medeniyettir.

* Doğu Akdeniz ve Batı Afrika sahillerinde ticaret kolonileri kurdular. Doğu ve Batı medeniyetlerinin kaynaşmasında taşıyıcı bir rol oynadılar.

* Mezopotamya çivi yazısından ve Mısır hiyeroglifinden etkilenecek HAREF YAZISI'nı (alfabe) buldular.

** Fenikeliler'in 22 harften oluşan yazıları, Yunanlılar'a, onlardan da Romalılar'a geçerek bugünkü Latin alfabesini oluşturmuştur.

* CAM'ı icat etmişler, Fildişi işlemeciliğinde ileri gitmişlerdir.

İBRANİ MEDENİYETİ

MÖ. 1500'lerde Filistin ve Lübnan dolaylarında yaşayan İbraniler, Sami ırkındandırlar.

* Hz. Musa zamanında birlik haline geldiler, devlet haline gelmeleri Hz. Davud zamanında oldu. En güçlü dönemleri Hz. Süleyman zamanıdır.

* Hz. Süleyman'dan sonra İbrani Devleti İsrail ve Yahudi Devleti olmak üzere ikiye ayrılmıştır. İsrail devletine Asurlular, Yahudi (Yuda) devletine ise Babilliler son vermişlerdir.

* Dinleri tek tanrılıdır. (Yahudilik=Musevilik). İlk çağın tek tanrılı dine inanan ilk kavmidir. Kutsal kitapları Tevrat'dır.

** İbraniler, Museviliği milli bir din olarak kabul ettiklerinden bu din diğer kavimler arasında fazla yayılmamıştır.

** Dinlerinin etrafında milli bir birlik oluşturduklarından dünyanın dört bir yanına dağılmış olmalarına rağmen birbirleriyle dayanışma içinde olmuşlardır.

* II. Dünya Savaşı sonunda İngiltere ve Amerika'nın yardımıyla bugünkü Filistin'de İsrail devletini kurmuşlardır.

* En önemli eserleri Kudüs'teki Mescid-i Aksa (Süleyman Mabedi)'dir.

APPENDIX-5 continues

Fenike Medeniyeti - İbrani Medeniyeti

Mixed Test - 12.02.2019

FENİKE MEDENİYETİ

(1) Lübnan Dağları ile sahili arasındaki bölgede yaşamış, gemicilik ve ticaretle gelişmiş bir medeniyettir.

FENİKE MEDENİYETİ

(2) ve Batı Afrika sahillerinde ticaret kolonileri kurdular.

- A) Doğu Trakya
- B) Doğu Akdeniz
- C) Güney Marmara
- D) Batı Trakya

FENİKE MEDENİYETİ

(3) Mezopotamya çivi yazısından ve Mısır hiyeroglifinden etkilenecek HARF YAZISI'ni (alfabe) buldular.

- A) True
- B) False

FENİKE MEDENİYETİ

(4) Hititler'in 22 harften oluşan yazıları, Yunanlılar'a, onlardan da Romalılar'a geçerek bugünkü Latin alfabesini oluşturmuştur.

- A) True
- B) False

FENİKE MEDENİYETİ

(5) CAM'ı icat etmişler, Fildişi işlemeciliğinde ileri gitmemişlerdir.

- A) True
- B) False

İBRANİ MEDENİYETİ

(6) MÖ. 1500'lerde Filistin ve dolaylarında yaşayan İbraniler, Sami ırkındandırlar.

- A) Yemen
- B) Lübnan
- C) Tunus
- D) Pakistan

İBRANİ MEDENİYETİ

(7) Hz. Musa zamanında birlik haline geldiler, devlet haline gelmeleri Hz. Muhammed zamanında oldu.

- A) True
- B) False

APPENDIX-5 continues

İBRANİ MEDENİYETİ

(8) En güçlü dönemleri zamanıdır.

- A) Hz. Muhammed
- B) Hz. Süleyman
- C) Zeynelabidin
- D) Hz. Ali

İBRANİ MEDENİYETİ

(9)'dan sonra İbrani Devleti İsrail ve Yahudi Devleti olmak üzere ikiye ayrılmıştır.

İBRANİ MEDENİYETİ

(10) devletine Asurlular, Yahudi (Yuda) devletine ise Babilliler son vermişlerdir.

- A) İran
- B) Afganistan
- C) İsrail
- D) Tunus

İBRANİ MEDENİYETİ

(11) (Yahudilik=.....).

- A) Vehhabilik
- B) Musevilik
- C) Müslümanlık
- D) Zerdüştlük

İBRANİ MEDENİYETİ

(12) İbraniler, Museviliği milli bir din olarak kabul ettiklerinden bu din diğer kavimler arasında fazla yayılmıştır.

- A) True
- B) False

İBRANİ MEDENİYETİ

(13) Dinlerinin etrafında milli bir birlik oluşturdıklarından dünyanın dört bir yanına dağılmış olmalarına rağmen birbirleriyle dayanışma içinde olmuşlardır.

- A) True
- B) False

İBRANİ MEDENİYETİ

(14) II. Dünya Savaşı sonunda İngiltere ve Kanada'nın yardımıyla bugünkü Filistin'de İsrail devletini kurmuşlardır.

- A) True
- B) False

İBRANİ MEDENİYETİ

(15) En önemli eserleri Kabil'teki Mescid-i Aksa (Süleyman Mabedi)'dir.

- A) True
- B) False