

# **STATISTICAL MODELING OF AGGLUTINATIVE LANGUAGES**

**A DISSERTATION SUBMITTED TO  
THE DEPARTMENT OF COMPUTER ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCE  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY**

By

**Dilek Z. Hakkani-TUR**

August, 2000

THESIS

QA

75.9

.N38

H35

2000



# STATISTICAL MODELING OF AGGLUTINATIVE LANGUAGES

A DISSERTATION SUBMITTED TO  
THE DEPARTMENT OF COMPUTER ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCE  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

By  
Dilek Z. Hakkani-Tür  
August, 2000

QA

76.9

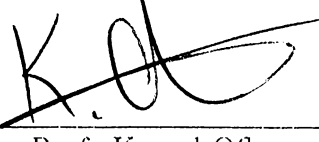
-N38

H35

2000

B053051

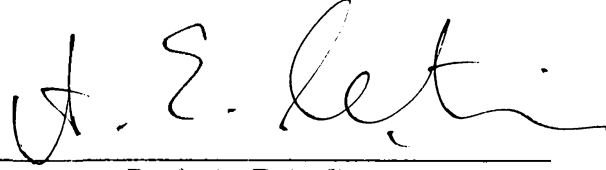
I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a dissertation for the degree of doctor of philosophy.



---

Assoc. Prof. Kemal Oflazer (Advisor)

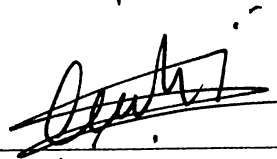
I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a dissertation for the degree of doctor of philosophy.



---

Prof. A. Enis Çetin

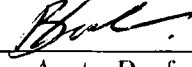
I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a dissertation for the degree of doctor of philosophy.



---

Asst. Prof. İlyas Çiçekli

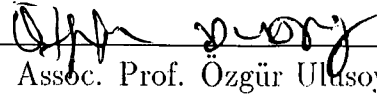
I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a dissertation for the degree of doctor of philosophy.



---

Asst. Prof. Bilge Say

I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a dissertation for the degree of doctor of philosophy.



---

Assoc. Prof. Özgür Ulusoy

Approved for the Institute of Engineering and Science:



---

Prof. Mehmet Baray  
Director of the Institute

ABSTRACT

STATISTICAL MODELING OF AGGLUTINATIVE  
LANGUAGES

Dilek Z. Hakkani-Tür  
Ph.D. in Computer Engineering  
• Supervisor: Assoc. Prof. Kemal Oflazer  
August, 2000

Recent advances in computer hardware and availability of very large corpora have made the application of statistical techniques to natural language processing a possible, and a very appealing research area. Many good results have been obtained by applying these techniques to English (and similar languages) in parsing, word sense disambiguation, part-of-speech tagging, and speech recognition. However, languages like Turkish, which have a number of characteristics that differ from English have mainly been left unstudied. Turkish presents an interesting problem for statistical modeling. In contrast to languages like English, for which there is a very small number of possible word forms with a given root word, for languages like Turkish or Finnish with very productive agglutinative morphology, it is possible to produce thousands of forms for a given root word. This causes a serious data sparseness problem for language modeling.

This Ph.D. thesis presents the results of research and development of statistical language modeling techniques for Turkish, and tests such techniques on basic applications of natural language and speech processing like morphological disambiguation, spelling correction, and  $n$ -best list rescoring for speech recognition. For all tasks, the use of units smaller than a word for language modeling were tested in order to reduce the impact of data sparsity problem. For morphological disambiguation, we examined  $n$ -gram language models and maximum entropy models using inflectional groups as modeling units. Our results indicate that using smaller units is useful for modeling languages with complex morphology and  $n$ -gram language models perform better than maximum entropy models. For  $n$ -best list rescoring and spelling correction, the  $n$ -gram language models that were developed for morphological disambiguation, and their approximations, via prefix-suffix models were used. The prefix-suffix models performed very well for  $n$ -best list rescoring, but for spelling correction, they could not beat word-based

models, in terms of accuracy.

”

*Keywords:* Natural Language Processing, Statistical Language Modeling, Agglutinative Languages, Morphological Disambiguation, Speech Recognition, Spelling Correction,  $n$ -gram Language Models, Maximum Entropy Models.

## ÖZET

# SONDAN EKLEMELİ DİLLERİN İSTATİSTİKSEL MODELLENMESİ

Dilek Z. Hakkani-Tür

Bilgisayar Mühendisliği, Doktora

Tez Yöneticisi: Doç. Dr. Kemal Oflazer

Ağustos, 2000

Bilgisayar donanımındaki yeni gelişmeler ve çok büyük derlemlerin varlığı istatistiksel tekniklerin doğal dil işlemeye uygulanmasını mümkün ve çok çekici bir araştırma alanı yapmıştır. Bu tekniklerin İngilizce ve benzeri dillerde cümle çözümleme (parsing), kelime anlamı tekleştirme (word sense disambiguation), sözcük sınıfı işaretleme (POS tagging) ve konuşma tanımaya uygulanmasıyla oldukça iyi sonuçlar elde edilmiştir. Ancak, Türkçe gibi, İngilizce ve benzeri dillerden bir takım farklı özellikleri olan diller genellikle bu açıdan incelenmemişlerdir. Türkçe'nin istatistiksel modellenmesi ilginç bir problemdir. Verilen bir kökten az sayıda kelime üretilen İngilizce ve benzeri dillerin aksine Türkçe ve Fince gibi üretken eklemeli biçimbirimi olan dillerde, verilen bir kökten binlerce, hatta milyonlarca, yeni kelime üretmek mümkündür. Bu dil modelleme açısından çok ciddi bir veri yetersizliği problemine sebep olur.

Bu doktora tezinde, Türkçe için istatistiksel dil modelleme tekniklerinin geliştirilmesi ve uygulanması ve bu tekniklerin biçimbirimsel tekleştirme, yazım hatalarının düzeltilmesi ve konuşma tanıma için aday ( $n$ -best) listesini yeniden değerlendirme gibi temel doğal dil ve konuşma işleme uygulamalarında denenmesi anlatılmaktadır. Bütün bu uygulamalarda veri yetersizliği probleminin etkisini azaltmak için kelimedenden daha küçük birimler kullanıldı. Biçimbirimsel tekleştirme için, çekim eki grupları (inflectional groups) modelleme birimi olarak kullanılarak  $n$ -birimli dil modelleri ( $n$ -gram language models) ve maksimum düzensizlik (maximum entropy) modelleri geliştirildi. Aldığımız sonuçlar, karmaşık biçimbirimsel yapıya sahip dilleri modellemek için sözcükten daha küçük birimler kullanmanın gerçekten de çok faydalı olduğunu gösterdi ve  $n$ -birimli dil modelleme yöntemi, maksimum düzensizlik yönteminden daha iyi sonuçlar verdi. Aday listesini yeniden değerlendirmek ve yazım hatalarının düzeltilmesi içinse biçimbirimsel tekleştirme için geliştirilen bu modeller ve bunların önek-sonek (prefix-suffix)



modelleri gibi yakınsamaları kullandı. Önek-sonек modelleri, aday listesinin yeniden değerlendirilmesinde çok iyi sonuçlar verdi, ancak yazım hatalarının düzeltilmesinde doğruluk açısından sözcük tabanlı modellerden daha iyi sonuç vermedi.

•

*Anahtar sözcükler:* Doğal Dil İşleme, İstatistiksel Dil Modelleme, Biçimbirimsel Tekleştirme, Konuşma Tanıma, Yazım Hatalarının Düzeltilmesi,  $n$ -birimli Dil Modelleri, Maksimum Düzensizlik Modelleri.

## Acknowledgments

I would like to express my deep gratitude to my supervisor Kemal Oflazer for his guidance, suggestions, invaluable encouragement and friendship throughout the development of this thesis. I feel really lucky for having worked with him.

I would like to thank Bilge Say, İlyas Çiçekli, and Özgür Ulusoy for reading and commenting on this thesis.

During my PhD study, I visited SRI International Speech Technology and Research Labs. I would like to thank Elizabeth Shriberg, Andreas Stolcke, and Kemal Sönmez for their helpful discussions and motivating suggestions. I would like to thank Andreas Stolcke for also providing us with the language modeling toolkit.

I would like to thank members of Center for Language and Speech Processing at Johns Hopkins University, Frederick Jelinek, David Yarowsky and Eric Brill for introducing me to statistical language modeling, during my visit to Johns Hopkins University.

I am grateful to my colleagues and friends Yücel Saygın, Kurtuluş Yorulmaz, Umut Topkara, Lidia Mangu, Madelaine Plauché, Denizhan Alparslan, Nihan Özyürek Bıçakçı, and many others who are not mentioned here by name.

My biggest gratitude is to my family. I am grateful to my parents and sister for their infinite moral support and help throughout my life. I thank my wonderful husband Gökhan for his steady support, encouragement, and love throughout difficult times in my graduate years.

To my family

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Motivation . . . . .	2
1.2.1	Statistical Language Modeling . . . . .	3
1.2.2	Turkish	4
1.3	Approach	5
1.4	Layout of the Thesis	6
<b>2</b>	<b>Statistical Language Modeling</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Evaluating the Performance of Models	8
2.2.1	Entropy	9
2.2.2	Cross Entropy . . . . .	9
2.2.3	Perplexity . . . . .	10
2.3	$n$ -gram Language Models . . . . .	10



2.4	Smoothing Techniques . . . . .	12
2.4.1	Deleted Interpolation . . . . .	13
2.4.2	Backing Off . . . . .	13
2.4.3	Good-Turing Smoothing . . . . .	14
2.5	Hidden Markov Models . . . . .	15
2.5.1	Finding the Best Path . . . . .	16
2.5.2	Finding the Probability of an Observation . . . . .	19
2.5.3	Parameter Estimation . . . . .	19
2.5.4	Using HMMs for Statistical Language and Speech Processing . . . . .	20
2.6	Maximum Entropy Models . . . . .	20
2.6.1	The Maximum Entropy Principle . . . . .	22
2.6.2	Representing Information via Features . . . . .	22
2.6.3	An Example . . . . .	23
2.6.4	Conditional Maximum Entropy Models . . . . .	25
2.6.5	Combining Information Sources . . . . .	26
2.6.6	Parameter Estimation . . . . .	26
<b>3</b>	<b>Turkish</b> . . . . .	<b>29</b>
3.1	Introduction . . . . .	29
3.2	Syntactic Properties of Turkish . . . . .	29
3.2.1	Word Order . . . . .	29

3.2.2	Morphology . . . . .	30
3.3	Issues for Language Modeling of Turkish	31
3.4	Examples of Morphological Ambiguity	35
3.5	Inflectional Groups	36
3.6	Statistics on the Inflectional Groups . . . . .	37
<b>4</b>	<b>Related Work</b>	<b>40</b>
4.1	Introduction . . . . .	40
4.2	Part-of-Speech Tagging . . . . .	40
4.3	Statistical Language Modeling . . . . .	42
4.4	Turkish	43
<b>5</b>	<b>Morphological Disambiguation</b>	<b>45</b>
5.1	Introduction . . . . .	45
5.2	Simplifying the problem for Turkish . . . . .	47
5.3	Morphological Disambiguation of Turkish with $n$ -gram Language Models . . . . .	48
5.3.1	Using IGs for Morphological Disambiguation . . . . .	49
5.3.2	An Example . . . . .	54
5.3.3	Implementation of the Models . . . . .	56
5.3.4	Experiments and Results . . . . .	59

5.4	Morphological Disambiguation of Turkish with Maximum Entropy Models . . . . .	64
5.4.1	The Probability Model . . . . .	65
5.4.2	Features for Morphological Disambiguation . . . . .	67
5.4.3	Training the Models . . . . .	68
5.4.4	Testing the Models . . . . .	69
5.4.5	Experiments and Results . . . . .	69
5.5	Discussion . . . . .	73
6	<b>Application to Speech Recognition</b> . . . . .	<b>76</b>
6.1	Introduction . . . . .	76
6.2	The Recognizer . . . . .	77
6.3	Problem . . . . .	78
6.4	Approximating the Acoustic Model Probabilities . . . . .	78
6.4.1	Equal Probabilities . . . . .	80
6.4.2	Linearly Decreasing Probabilities . . . . .	80
6.4.3	Exponentially Decreasing Probabilities . . . . .	80
6.5	Language Modeling . . . . .	82
6.5.1	Word-Based Language Modeling . . . . .	82
6.5.2	IG-Based Language Modeling . . . . .	83
6.5.3	Prefix-Suffix Language Modeling . . . . .	85

6.6	Experiments and Results . . . . .	86
6.6.1	Training and Test Data . . . . .	87
6.6.2	Initial Performance of the Recognizer . . . . .	88
6.6.3	Word-Based Language Modeling . . . . .	88
6.6.4	IG-Based Language Modeling . . . . .	89
6.6.5	Prefix-Suffix Language Modeling . . . . .	90
6.7	Discussion . . . . .	92
<b>7</b>	<b>Application to Spelling Correction . . . . .</b>	<b>94</b>
7.1	Introduction . . . . .	94
7.2	Spelling Errors . . . . .	95
7.3	Language Modeling . . . . .	96
7.4	Experiments and Results . . . . .	97
7.4.1	Training Data . . . . .	97
7.4.2	Test Data . . . . .	97
7.4.3	Results . . . . .	98
7.5	Discussion . . . . .	100
<b>8</b>	<b>Conclusions and Future Work . . . . .</b>	<b>101</b>
8.1	Summary . . . . .	101
8.2	Contributions . . . . .	102
8.2.1	Theoretical Contributions . . . . .	102



8.2.2	Experimental Contributions	103
8.2.3	Contributions for Further Studies on Turkish . . . . .	103
8.3	Future Work . . . . .	104
8.3.1	Real Stem-Suffix Models . . . . .	104
8.3.2	Class-based Models . . . . .	105
8.3.3	Automatic Acquisition of Language Modeling Units . . . . .	106
<b>A</b>	<b>Categories</b>	<b>117</b>
A.1	Major Part-of-Speech . . . . .	117
A.2	Minor Part-of-Speech . . . . .	119
A.3	Agreement . . . . .	121
A.4	Possessive . . . . .	121
A.5	Case . . . . .	122
A.6	Polarity	122
A.7	Tense-Aspect-Mood Marker 1	122
A.8	Tense-Aspect-Mood Marker 2	123
A.9	Copula . . . . .	123

# List of Figures

1.1	A generalization of all the tasks. . . . .	5
2.1	A 4-gram language model. . . . .	11
2.2	A simple arc-emission HMM and computation of the probability of an observation using a trellis. . . . .	17
2.3	A fragment of the Markov Model for a bigram language model. . .	21
2.4	A fragment of the HMM for estimating the parameters of a linearly interpolated bigram language model. . . . .	21
3.1	The list of words that can be obtained by suffixing only one morpheme to a noun 'masa' ('table' in English.). . . . .	32
3.2	Example of possible word formations with derivational and inflectional suffixes from a Turkish verb. . . . .	33
3.3	Inflectional groups in a word and the syntactic relation links. . . .	37
3.4	An example dependency tree for a Turkish sentence. The words are segmented along the IG boundaries. . . . .	38
5.1	The dependency between current word and its history word according to Model 1. . . . .	51

5.2	The dependency between current word and its history word according to Models 2 and 3. . . . .	52
5.3	Implementation of the $n$ -gram models.	57
5.4	The trigram HMM for morphological disambiguation. $\langle S \rangle$ is the sentence start tag, and $\langle /S \rangle$ is the sentence end tag. . . . .	58
6.1	The $n$ -best list rescoring process.	79
6.2	Approximating acoustic model probabilities with a linear function.	81
6.3	Approximating acoustic model probabilities with an exponential function. . . . .	81
6.4	Dependence of the letter sequences. . . . .	86

# List of Tables

2.1	A simple bigram example for Good-Turing smoothing.	15
3.1	The number of possible word formations obtained by suffixing 1, 2 and 3 morphemes to a NOUN, a VERB and an ADJECTIVE.	31
3.2	Vocabulary sizes for two Turkish and English corpora. . . . .	34
3.3	The perplexity of Turkish and English corpora using word-based trigram language models. . . . .	34
3.4	Numbers of Tags and IGs	38
5.1	Accuracy results for different models. In the first column, US is an abbreviation for unambiguous sequences.	62
5.2	The contribution of the individual models for the best case. . . . .	63
5.3	Examples for representations of IGs using 9 categories. The category that is missing in the IG takes the value “-”.	66
5.4	The accuracy results with models trained varying the counts of the features and IGs to include.	71
5.5	The number of features used for the models, with different threshold value for the features and IGs to include. . . . .	71



5.6	The accuracy results for a threshold weight of 0.1. “all” includes 9 models. The test set used for these experiments is test2.	72
5.7	The accuracy results for a threshold weight of 0.01. “all” includes the remaining 6 models.	72
5.8	The number of type-2 features used for the models, with IG Count>5 and Feature Count Threshold $\geq$ 1. . . . .	73
5.9	The accuracy results with Type-2 features. Only the IG bigrams that occurred more than 5 times were used when finding the features.	73
6.1	The initial performance of the recognizer. . . . .	88
6.2	The performance of the recognizer after rescoring the $n$ -best list with a word based LM. . . . .	89
6.3	The performance of the recognizer after rescoring the $n$ -best list with an IG based LM. The AM Weight stands for the Acoustic Model Weight. . . . .	90
6.4	The performance of the recognizer after rescoring the $n$ -best list with a prefix-suffix LM, which has a root size of 3 letters and suffix size of 2 letters. . . . .	90
6.5	The performance of the recognizer after rescoring the $n$ -best list with a prefix-suffix LM, which has a root size of 4 letters and suffix size of 2 letters. . . . .	91
6.6	The performance of the recognizer after rescoring the $n$ -best list with a prefix-suffix LM, which has a root size of 4 letters and suffix size of 3 letters. . . . .	91
6.7	The effect of exponentially decreasing acoustic model probabilities on performance. . . . .	92

*LIST OF TABLES*

xx

7.1	Examples of 4 types of spelling errors for English and Turkish. . .	96
7.2	The variations of our test data.	98
7.3	The accuracy results for the spelling correction.	99

"

# Chapter 1

## Introduction

### 1.1 Overview

Statistical language modeling is the study of finding, characterizing and exploiting the regularities in natural language using statistical techniques. Recent advances in computer hardware, and availability of very large corpora have made the application of statistical techniques to natural language processing a feasible and a very appealing research area. Many useful and successful results have been obtained by applying these techniques to English (and similar languages) in parsing, word sense disambiguation, part-of-speech tagging, speech recognition, etc. However, languages which display a substantially different behavior than English, like Turkish, Czech, Hungarian, etc. in that, they have agglutinative or inflecting morphology and relatively free constituent order, have mainly been left unstudied.

This thesis presents our work on the development and application of statistical language modeling techniques for Turkish, and testing such techniques on basic applications of natural language processing like morphological disambiguation,  $n$ -best list rescoring for speech recognition, and spelling correction.

Morphological disambiguation is the problem of selecting the sequence of morphological analyses (including the root) corresponding to a sequence of words, from the set of possible parses for these words. Morphological disambiguation is a very important step in natural language understanding, text-to-speech synthesis, etc. For example, the pronunciation of the words may differ according to their parses (i.e., the Turkish word 'bostancı' is pronounced differently depending on whether it is a proper noun (the name of a location), or it is a common noun). Morphological disambiguation also reduces the search space during syntactic parsing [Voutilainen, 1998].

Speech recognition is the task of finding the uttered sequence of words, given the corresponding acoustic signal. Most of the time, the recognizer outputs a list of candidate utterances, that is, an  $n$ -best list. Using a language model and the  $n$ -best list, the accuracy of the speech recognizer can be improved. This process is called  $n$ -best list rescoring, and is a very important step for improving the speech recognition accuracy.

Spelling correction is the task of finding the correct version of a mis-spelled word, among the candidates that the spell checker proposes. Spelling checkers and correctors are a part of all modern word processors and are also important in applications like optical character recognition and hand writing recognition.

The techniques developed in this thesis comprise the first comprehensive use of statistical modeling techniques for Turkish, and they can be used for other language processing and understanding, and speech processing tasks. These techniques can certainly be applicable to other agglutinative languages with productive derivational morphology.

## 1.2 Motivation

The motivation for this thesis work can be summarized as below:



- Statistical language modeling techniques are successfully used for natural language processing tasks, for languages like English.
- Turkish displays different characteristics than mostly studied languages like English. Turkish is a free-constituent order language, with an agglutinative morphology. These differences complicate the straightforward application of statistical language modeling techniques to Turkish.
- There have been no previous studies in statistical language modeling of Turkish.

In the following subsections, we will concentrate on the advantages of statistical language processing techniques, and the differences of Turkish from other mostly studied languages, which motivate this study.

### 1.2.1 Statistical Language Modeling

Approaches to speech and language processing can be divided into two main paradigms: *Symbolic* and *Statistical*. Symbolic approaches are based on hand-crafted linguistically motivated rules. This paradigm is rooted back to Chomsky's work on formal language theory, and has become very popular in linguistics and computer science. On the other hand, statistical approaches attempt to learn the patterns of the language using training data. Statistical language processing emerged from the electrical engineering domain, by the application of Bayesian method to the problem of optical character recognition [Jurafsky and Martin, 1999]. Statistical methods are based on probability theory, statistics, and information theory.

Some of the most important advantages and disadvantages of these approaches are listed below. Note that, the advantage of one approach is generally the disadvantage of the other.

- Symbolic approaches are usually developed for specific domains, and require extensive labor in building the rules or the grammars. Changes in the

specifications of the task result in expensive tuning.

- Statistical approaches generally require annotated training data, which is usually unavailable. This is true especially for lesser studied languages, such as Turkish.
- For most of the tasks, rule-based systems give better performance than statistical systems. However, in recent years, with the availability of larger training data and more sophisticated statistical frameworks, it is possible to get better results using statistical methods.
- Statistical methods are more suitable for combining multiple information sources, such as prosodic or linguistic information.

### 1.2.2 Turkish

Turkish is a free constituent order language, in which constituents at certain phrase levels can change order rather freely according to the discourse context and text flow. The typical order of the constituents is Subject-Object-Verb, but other orders are also common, especially in discourse. The morphology of Turkish enables morphological markings on the constituents to signal their grammatical roles without relying on the word order. This doesn't mean that word order isn't important, sentences with different word orders reflect different pragmatic conditions. However, the free constituent order property complicates the statistical language modeling approach.

Turkish has agglutinative morphology, with productive inflectional and derivational suffixations. Hence, the number of distinct Turkish word forms is very large. So, we have to deal with data sparseness problem while training our language models. A detailed discussion on the properties of Turkish is given in the following chapters.

### 1.3 Approach

All of the tasks, to which we applied our techniques, search for a sequence among possible sequences, using statistical language modeling techniques. So, all of these problems can be represented as the problem of finding the most probable sequence of units,  $X^*$ , among the set of possible sequences of units,  $\mathcal{X}$ , given corresponding information,  $Y$ . Then the problem can be represented as follows:

$$X^* = \operatorname{argmax}_{X \in \mathcal{X}} P(X|Y)$$

In morphological disambiguation,  $X$  is the sequence of morphological parses and  $Y$  is the sequence of words that we are trying to analyze morphologically. In  $n$ -best list rescoring,  $X$  is the sequence of words and  $Y$  is the sequence of acoustic signals that we are trying to transcribe. In spelling correction,  $X$  is again the sequence of words and  $Y$  is the sequence of possibly mis-typed words.

Figure 1.1 shows the general architecture for all these tasks. The decoder is the morphological analyzer for morphological disambiguation, speech recognizer for  $n$ -best list rescoring, and the spelling checker for spelling correction. All of these systems output a set of possible candidates, and we use statistical models to select one of these possible candidates, which is shown as the rescoring box in that figure.

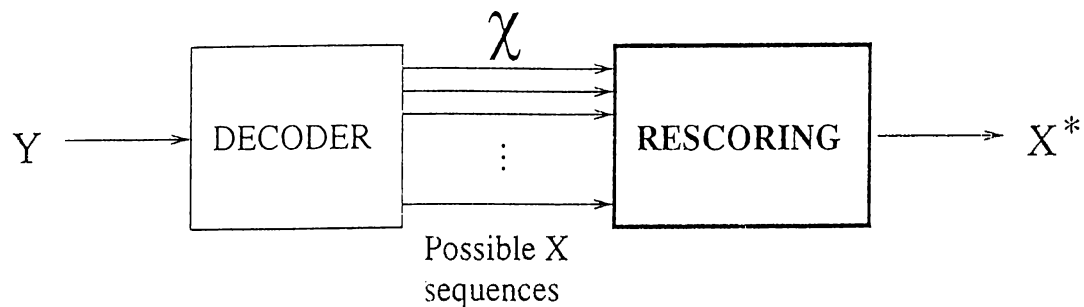


Figure 1.1: A generalization of all the tasks.

## 1.4 Layout of the Thesis

The organization of this thesis is as follows: In Chapter 2, we describe the basics of statistical language modeling techniques. In Chapter 3, we summarize the properties of Turkish, which make it different from languages like English, which have been amply studied in this context, emphasizing the properties which complicate the straightforward application of statistical language modeling techniques. In Chapter 4, we describe briefly the related work on part-of-speech tagging, morphological disambiguation, statistical language modeling and Turkish. We include the related work on part-of-speech tagging, since our models are influenced from statistical part-of-speech tagging studies for English. In Chapter 5, we describe two approaches for morphological disambiguation of Turkish, based on  $n$ -gram language models and maximum entropy models. We present and compare our results with both techniques. In Chapters 6 and 7, we describe the application of our  $n$ -gram based models and their approximation to speech recognition and spelling correction, respectively. We present our ideas for future work and conclude in Chapter 8.

# Chapter 2

## Statistical Language Modeling

### 2.1 Introduction

Statistical language modeling is the study of the regularities in the natural language, and capturing them in a statistical model. In this framework, natural language is viewed as a stochastic process, and the units of text (i.e., letters, morphemes, words, sentences) are seen as random variables with some probability distribution. Statistical language modeling attempts to capture local grammatical regularities.

Traditionally, statistical language modeling has been extensively used in speech recognition systems [Bahl *et al.*, 1983; Sankar *et al.*, 1998; Beyerlein *et al.*, 1998, among others]. For example, in speech recognition, given an acoustic signal  $A$ , the aim is to find the corresponding sequence of words  $W$ . So, we seek the word sequence  $W^*$  that maximizes  $P(W|A)$ . Applying Bayes' Law, we get:<sup>1</sup>

$$W^* = \operatorname{argmax}_W P(W|A) = \operatorname{argmax}_W \frac{P(W) \times P(A|W)}{P(A)} \quad (2.1)$$

---

<sup>1</sup>

$\operatorname{argmax}_x f(x)$  is the value of  $x$  that maximizes  $f(x)$ .

The above maximization is carried out with the variable  $A$  fixed, so  $P(A)$  is constant for different  $W$ , which leaves us with the following equation:

$$W^* = \operatorname{argmax}_W P(W|A) = \operatorname{argmax}_W P(W) \times P(A|W) \quad (2.2)$$

For a given acoustic signal  $A$ ,  $P(A|W)$  is estimated by the acoustic model, and  $P(W)$  is estimated by the language model. So, language modeling deals with assigning a probability to every conceivable word string,  $W$ . The probability distribution of the units of a statistical model is inferred using on-line text and speech corpora.

We can formulate many problems of natural language processing, like morphological disambiguation and noun phrase extraction, in a similar framework. For example, in morphological disambiguation, the input is a sequence of words instead of the acoustic signal, and the aim is to find the sequence of morphological parses belonging to each word. In the case of noun phrase bracketing, the output is the type of the boundary between the words, which can also be seen as tags attached to the words preceding or following the boundary.<sup>2</sup>

## 2.2 Evaluating the Performance of Models

The most common metrics to evaluate the performance of language models are *entropy*, *cross entropy* and *perplexity*. The concept of entropy was borrowed from thermodynamics by Shannon [Shannon, 1948], as a way of measuring the information capacity of a channel, or the information content of a language. Another measure, especially used by the speech recognition community is perplexity [Bahl *et al.*, 1983]. In the following subsections, we will briefly describe each of these metrics. For a more detailed discussion on these metrics, the reader is referred to one of the textbooks on statistical language modeling, such as the one by Manning and Schütze [Manning and Schütze, 1999].

---

<sup>2</sup>The type of the boundary between the words can be “beginning of noun phrase”, “end of noun phrase”, or “none”.

### 2.2.1 Entropy

*Entropy* is a measure of average uncertainty of a random variable [Cover and Thomas, 1991]. Let  $X$  be a random variable that ranges over the unit we are predicting, like words or letters, and  $P(x)$  be the probability mass function of the random variable  $X$  over the alphabet of our units:

$$P(x) = P(X = x), x \in X \quad (2.3)$$

Then the entropy of this random variable, denoted by  $H(X)$  is:

$$H(X) = - \sum_{x \in X} P(x) \log_2 P(x) \quad (2.4)$$

The log can be computed in any base. If we use base 2, then the entropy is measured in bits.

Entropy, the amount of information in a random variable, is the lower bound on the average number of bits it would take to encode the outcome of that random variable.

### 2.2.2 Cross Entropy

The quality of a language model  $M$  can be judged by its *cross entropy* [Charniak, 1993; Manning and Schütze, 1999]:

$$H(T, M) = - \frac{1}{n} \sum_{w_1^n \in L} P_T(w_1^n) \log P_M(w_1^n) \quad (2.5)$$

where  $w_1^n = w_1, w_2, \dots, w_n$  is a sequence of words of the language  $L$ ,  $P_T$  is the actual probability distribution that generated the data, that is, the possible word sequences of the language in consideration, and  $P_M$  is a model of  $P_T$ , that is, an approximation to  $P_T$ , that we try to construct using training data. According to the Shannon-McMillan-Breiman Theorem [Cover and Thomas, 1991], if language is both stationary and ergodic, the following equations hold:<sup>3</sup>

---

<sup>3</sup>A stochastic process is *stationary*, if the probabilities that it assigns to a sequence are invariant with respect to time changes. A language is *ergodic*, if any sample of the language, if made long enough, is a perfect sample.

$$H(T, M) = - \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{w_1^n} P_T(w_1^n) \log P_M(w_1^n) \quad (2.6)$$

$$= - \lim_{n \rightarrow \infty} \frac{1}{n} \log P_M(w_1^n) \quad (2.7)$$

Cross entropy is a measure of how much our approximated probability distribution,  $M$ , departs from actual language use, so one of the goals in language processing is to minimize it. Cross entropy can also be used to compare different probabilistic models, the model that has a lower cross entropy is better than the models that have higher cross entropies, in that it is closer to the actual probability distribution, that generates the language we use.

### 2.2.3 Perplexity

In the speech recognition community, often the *perplexity* of the data with regard to the model is reported to evaluate the performance of a language model [Manning and Schütze, 1999]:

$$\text{perplexity}(T, M) = 2^{H(T, M)} \quad (2.8)$$

A perplexity of  $k$  means that you are as surprised on the average, as you would have been, if you had to guess between  $k$  equiprobable choices at each step. So the aim is again to minimize perplexity.

## 2.3 $n$ -gram Language Models

Let  $W = w_1 w_2 \dots w_n = w_1^n$  be a sequence of words, where  $w_i$  are the words in an hypothesis.  $P(W)$  can be estimated using the chain rule:

$$P(W) = \prod_{i=1}^n P(w_i | w_1, w_2, \dots, w_{i-1}) = \prod_{i=1}^n P(w_i | w_1^{i-1}) \quad (2.9)$$

In any practical natural language processing system, even with a moderate vocabulary size, it is clear that the language model probabilities  $P(w_i | w_1^{i-1})$  can



not be stored for each possible sequence  $w_1 w_2 \dots w_i$ . One way of limiting the number of probabilities is to partition the possible word histories  $w_1 w_2 \dots w_i$  into a reasonable number of equivalence classes  $\Phi(w_1 w_2 \dots w_i)$ . An effective definition of equivalence classes is the conventional  $n$ -gram language model, where two sequences of words are considered equivalent if they end in the same  $n - 1$  words:

$$P(w_i | w_1^{i-1}) \approx P(w_i | w_{i-n+1}^{i-1}). \quad (2.10)$$

Figure 5.3 gives an example of a 4-gram model that approximates the probability of  $w_6$ , given all the previous words.

$w_1$	$w_2$	$w_3$	$w_4$	$w_5$	$w_6$
Orta	Asya'daki	petrol	ve	enerji	?
$P(w_6   \text{Orta, Asya'daki, petrol, ve, enerji}) \approx P(w_6   \text{petrol, ve, enerji})$					

Figure 2.1: A 4-gram language model.

$n$ -gram language models can be trained (that is, the probabilities  $P(w_i | w_{i-n+1}^{i-1})$  can be estimated) using a training corpus, and counting all the  $n$ -grams. The probability of a particular word  $w_n$ , given a sequence of  $n - 1$  words  $w_1^{n-1}$  is estimated as:

$$P(w_n | w_1^{n-1}) = \frac{C(w_1^{n-1}, w_n)}{\sum_{w \in W} C(w_1^{n-1}, w)} \quad (2.11)$$

where  $C(w_1^{n-1}, w_n)$  is the number of times the word sequence  $w_1^n$  occurred in the training text. This ratio is called a *relative frequency*, and the use of relative frequencies in order to estimate probabilities is an example of the technique known as Maximum Likelihood Estimation (MLE), since the resulting probability distribution is the one using which the likelihood of the training data is maximized [Jurafsky and Martin, 1999].

Even for small values of  $n$ , the number of probabilities to be estimated in an  $n$ -gram model is enormous. Consider a 3-gram language model. For a vocabulary of 20,000 words, the number of 3-word sequences, thus the number of probabilities to be estimated is  $8 * 10^{12}$ . This causes a data sparseness problem, since there is rarely enough data to estimate these probabilities. So, the  $n$ -grams that are

not seen in the training data are assigned a zero probability, some of which should really have a non-zero probability. These zero-probability  $n$ -grams can be assigned small probabilities using smoothing techniques. In the next section, we will briefly mention some of the most popular smoothing techniques.

There are methods for clustering the words that are similar (i.e., that occur in similar contexts) into *classes*, so that the vocabulary size is reduced to the number of classes [Brown *et al.*, 1992b; Martin *et al.*, 1995; McMahon and Smith, 1996]. As a result, the parameter space spanned by  $n$ -gram language models is also reduced, and the reliability of the estimates is increased.

The weakness of  $n$ -gram language models is that with this method, it is assumed that a word can only depend on the preceding  $n - 1$  words, although this is not always the case for natural language. For example, in the sentence “The dog that chased the cat barked.”, the history of the word ‘barked’ consists of the words ‘the’ and ‘cat’ in a 3-gram model. On the other hand,  $n$ -gram models have been surprisingly successful in many domains.

## 2.4 Smoothing Techniques

Smoothing is the process of assigning small probabilities to  $n$ -grams that were not seen in the training data because of data sparseness. Different smoothing methods usually offer similar performance results. Chen and Goodman [1996] present extensive evaluations of different smoothing algorithms and demonstrate that the performance of certain techniques depend greatly on the training data size and  $n$ -gram order (that is, the number  $n$ ). In the following subsections, we briefly describe some of the most popular smoothing techniques.

### 2.4.1 Deleted Interpolation

A solution to the sparse data problem is to interpolate multiple models of order  $1, \dots, n$ , so that

$$P(w_i|w_{i-n+1}^{i-1}) = \lambda_1 \times P(w_i|w_{i-n+1}^{i-1}) + \lambda_2 \times P(w_i|w_{i-n+2}^{i-1}) + \dots + \lambda_n \times P(w_i) \quad (2.12)$$

where  $\sum_{i=1}^n \lambda_i = 1$ , that is, we weight the contribution of each model so that the result is another probability function.

The values of the weights  $\lambda_i$  may be set by hand, but in order to find the weights that work best, usually a previously unseen corpus, called the *held-out data* is used. The values of  $\lambda_i$  that maximize the likelihood of that corpus are selected using the Expectation Maximization (EM) algorithm [Bahl *et al.*, 1983; Jelinek, 1998].

Linear interpolation can also be used as a way of combining multiple knowledge sources. Combining models using linear interpolation or another method can be seen as a solution to the blindness of the  $n$ -gram models to larger contexts, as well as the data sparseness problem.

### 2.4.2 Backing Off

The backoff smoothing technique, similar to deleted interpolation, uses lower order probabilities in case there is not enough evidence from higher order  $n$ -grams. But, instead of interpolating the models, this method backs off to a lower order model. Backoff  $n$ -gram modeling is a method introduced by Katz [1987]. For example, the trigram model probabilities, according to the backoff method, can be represented as follows:

$$P(w_i|w_{i-2}, w_{i-1}) = \begin{cases} P(w_i|w_{i-2}, w_{i-1}) & \text{if } C(w_{i-2}, w_{i-1}, w_i) > \epsilon_1 \\ \alpha_1 \times P(w_i|w_{i-1}) & \text{if } C(w_{i-2}, w_{i-1}, w_i) \leq \epsilon_1 \text{ and } C(w_{i-1}, w_i) > \epsilon_2 \\ \alpha_2 \times P(w_i) & \text{otherwise} \end{cases} \quad (2.13)$$

The values of  $\alpha_1$  and  $\alpha_2$  are chosen appropriately, so that  $P(w_i|w_{i-2}, w_{i-1})$  is normalized. In this way, when there is not enough evidence to estimate the probability using the trigram counts, we back off and rely on the bigrams.

### 2.4.3 Good-Turing Smoothing

Good-Turing methods provide a simple estimate of the probability of the objects not seen in the training data, as well as an estimation of the probabilities of observed objects, that is consistent with the total probability assigned to the unseen objects [Gale, 1994]. The basic idea is to re-estimate the probability values to assign to  $n$ -grams that do not occur or occur very rarely in the training data, by looking at the number of  $n$ -grams which occur more frequently.

Let  $r$  be a frequency in the training data, and  $N_r$  be the frequency of the frequency  $r$ , and  $N$  be the total number of objects observed (in our case, the size of the training data). So,  $N_5 = 11$  means that there are only 11 distinct  $n$ -grams which occurred 5 times in the training data. Let  $P_r$  be the probability that we estimate for the objects seen  $r$  times in the training data. Then, according to the Good-Turing methods,

$$P_r = \frac{r^*}{N}. \quad (2.14)$$

The  $r^*$  should be set in a way that makes the sum of the probabilities for all the objects equal to 1. A precise statement of the theorem underlying the Good-Turing Methods is [Gale, 1994]:

$$r^* = (r + 1) \frac{E(N_{r+1})}{E(N_r)} \quad (2.15)$$

where  $E(x)$  represents the expectation of the random variable  $x$ . Therefore, the total probability assigned to the unseen objects is  $E(N_1)/N$ . This method assumes that we already know the number of unseen  $n$ -grams. The number of unseen  $n$ -grams can be computed using the vocabulary size, and the number of seen  $n$ -grams, so this method assumes that we already know the vocabulary size. Table 2.4.3 gives a simple example of smoothing the bigram probabilities computed using a training text of 65 tokens, where the vocabulary size,  $V$  is 10.

$r$	$f$	$P$	$r^*$	$P_{smoothed}$
0	63	0	0.3077	0.00488
1	20	0.01538	1	0.01538
2	10	0.03077	1.2	0.01855
3	4	0.04615	2	0.03077

Table 2.1: A simple bigram example for Good-Turing smoothing.

The third column lists the probabilities before smoothing, and the fifth column lists the smooth probabilities. The mass probability reserved for unseen bigrams is 0.3077, and is distributed among unseen bigrams. The number of unseen bigrams is  $V^2 - N$ .

## 2.5 Hidden Markov Models

In this section, we will discuss the most widely used and the most successful technique in the speech recognition domain, the hidden Markov models (HMMs). HMMs are also widely used for other language processing tasks like part-of-speech tagging, information extraction, etc.

An HMM is a probabilistic finite state machine specified by a five-tuple  $M = \langle S, \Sigma, \Pi, T, O \rangle$ . Here  $S$  is the set of the states with a unique starting state  $s_0$ .  $\Sigma$  is the output alphabet.  $\Pi$  is the set of initial state probabilities,  $T$  is the set of state transition probabilities,  $p(s_i | s_{i-1})$ , and  $O$  is the set of output probabilities (either for transitions in *arc-emission HMMs*,  $q(w_i | s_{i-1}, s_i)$  or for states in *state-emission HMMs*,  $r(w_i | s_i)$ ) [Manning and Schütze, 1999]. The probability of observing an HMM output string  $w_1, w_2, \dots, w_n$  can be computed by summing the probabilities of all the paths that generate that string. Therefore, the probability of  $w_1, w_2, \dots, w_n$  is given by:

$$P(w_1, w_2, \dots, w_n | M) = \sum_{a_0, \dots, a_n} \prod_{k=1}^n p(a_k | a_{k-1}) \times q(w_k | a_{k-1}, a_k) \quad (2.16)$$

for arc-emission HMMs and

$$P(w_1, w_2, \dots, w_n | M) = \sum_{a_1, \dots, a_n} \prod_{k=1}^n p(a_k | a_{k-1}) \times r(w_k | a_k) \quad (2.17)$$

for state-emission HMMs, where  $a_k \in S$  represent the states traversed while emitting the output. These two HMM formulations are entirely equivalent [Jelinek, 1998], so we will only give the algorithms for arc-emission HMMs in the remaining of this chapter.

Figure 2.2 is an example of a three-state arc-emission HMM. The states are denoted by  $S_0$ ,  $S_1$ , and  $S_2$ . The arcs represent the transitions between states, and are shown by the lines and arrows. Each arc is marked by a pair  $x : y$ , where  $x$  is the symbol emitted if that arc is taken, and  $y$  is the probability of taking that transition and outputting  $x$ . The calculation of the observation probability of the sequence “baab” using its *trellis* is also shown in this figure. A trellis is an easy way of showing the time evolution of the traversal process [Jelinek, 1998]. The number of stages on the trellis is determined by the number of symbols in the output. There are two paths for generating “baab” as output, marked as solid lines on the trellis. The probability of generating this string is the sum of the probability of following those two paths.

### 2.5.1 Finding the Best Path

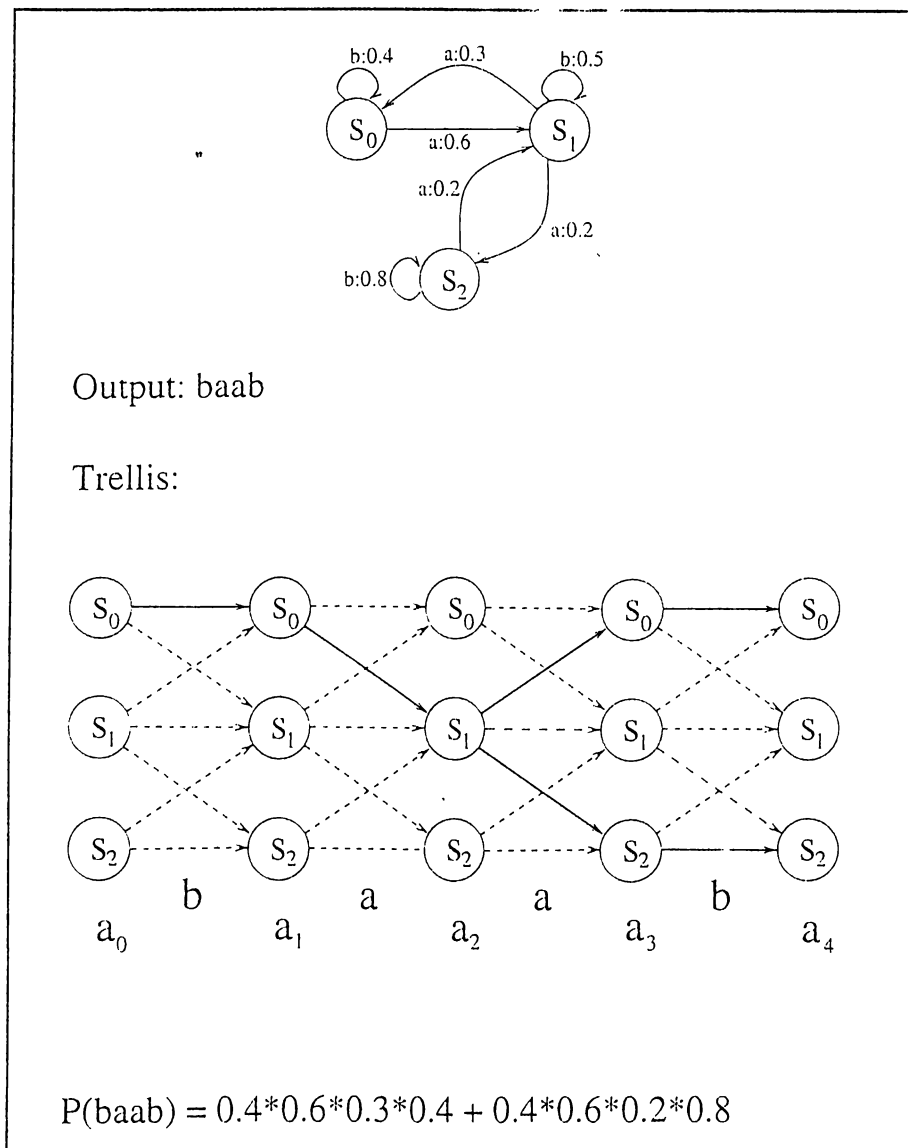
Given an observed output sequence  $W = w_1, w_2, \dots, w_n$ , and an HMM  $M = \langle S, \Sigma, \Pi, T, O \rangle$ , we can find the state sequence  $A^* = a_1^*, a_2^*, \dots, a_n^*, a_{n+1}^*$  most likely to have caused it, using the Viterbi algorithm, a dynamic programming algorithm [Viterbi, 1967]. Our aim is to find  $A^*$ , satisfying the following:

$$A^* = \underset{A}{\operatorname{argmax}} P(A | W, M) \quad (2.18)$$

$$= \underset{A}{\operatorname{argmax}} \frac{P(A, W | M)}{P(W | M)} \quad (2.19)$$

$$= \underset{A}{\operatorname{argmax}} P(A, W | M) \quad (2.20)$$

The probability  $P(W | M)$  is a constant for all  $A$ , since the sequence  $W$  is fixed, so it does not affect the result of the maximization. Define a variable,  $\delta_j(t)$ , which



The solid lines on the trellis are the transitions that are taken. The probability of the observation is found by summing the probability of the two paths that produce this observation.

Figure 2.2: A simple arc-emission HMM and computation of the probability of an observation using a trellis.

stores the probability of the most probable path which leads to that node, for each node in the trellis:

$$\delta_j(t) = \max_{a_1, \dots, a_{t-1}} P(a_1, a_2, \dots, a_{t-1}, w_1, w_2, \dots, w_{t-1}, a_t = j | M) \quad (2.21)$$

and define another variable,  $\Psi_j(t)$ , which stores the node of the incoming arc that led to this most probable path. The most probable path can be computed as follows:

1. *Initialize:*

$$\delta_j(1) = \pi_j \quad \text{for } 1 \leq j \leq N$$

where  $N$  is the number of states of the HMM.

2. *Induction:* Compute

$$\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) \times p(t_j | t_i) \times q(w_t | t_i, t_j) \quad (2.22)$$

for  $1 \leq j \leq N$ , and store the back-trace:

$$\Psi_j(t+1) = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(t) \times p(t_j | t_i) \times q(w_t | t_i, t_j) \quad (2.23)$$

3. *Termination:*

$$a_{n+1}^* = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(n+1) \quad (2.24)$$

$$P(W) = \max_{1 \leq i \leq N} \delta_i(n+1) \quad (2.25)$$

where  $P(W)$  is the probability of the most probable state sequence outputting  $W$ , and  $a_{n+1}^*$  is the final state of that sequence.

4. *Backtracking:* The state sequence,  $A^* = a_1^*, a_2^*, \dots, a_n^*, a_{n+1}^*$ , can be obtained by backtracking from state  $a_{n+1}^*$ .

$$a_t^* = \Psi_{t+1}(t+1), \quad t = n, \dots, 1.$$



### 2.5.2 Finding the Probability of an Observation

Given an observed output sequence  $W = w_1, w_2, \dots, w_n$ , and an HMM  $M = \langle S, \Sigma, \Pi, T, O \rangle$ , the probability of the observing a string can be computed by summing the probability of all the paths that generate that string, as mentioned in the previous sections. But, as in the problem of finding the best state sequence, there is no need to enumerate all the possible paths, and then sum their probability. There is a dynamic programming algorithm, very similar to Viterbi algorithm, that computes the probability of an observation sequence [Manning and Schütze, 1999].

Define a variable  $\alpha_j(t)$ , which stores the total probability of being in state  $t_j$  at time  $t$  (so the observations  $w_1, \dots, w_{t-1}$  were seen) for each node of the trellis. We can compute  $\alpha_j(t)$  by summing the probabilities of all possible ways of reaching that node:

$$\alpha_j(t) = \sum_{i=1}^N \alpha_i(t-1) \times p(t_j|t_i) \times q(w_t|t_i, t_j) \quad (2.26)$$

Therefore the algorithm is as follows:

1. *Initialize:*

$$\alpha_j(1) = \pi_j \quad \text{for } 1 \leq j \leq N$$

where  $N$  is the number of states of the HMM.

2. *Induction:* Compute

$$\alpha_j(t+1) = \sum_{i=1}^N \alpha_i(t) \times p(t_j|t_i) \times q(w_t|t_i, t_j) \quad (2.27)$$

3. *Termination:*

$$P(W|M) = \sum_{i=1}^N \alpha_i(n+1) \quad (2.28)$$

### 2.5.3 Parameter Estimation

There is no good way of estimating both the structure and the parameters of an HMM at the same time. But, if we design the HMM using our intuition and

knowledge of the situation, we can estimate the parameters of the HMM through the use of a special case of the Expectation Maximization (EM) algorithm, the Baum-Welch or Forward-Backward algorithm [Bahl *et al.*, 1983; Jelinek, 1998]. In the subsequent sections, instead of using the parameter estimation algorithms, we use the relative frequencies as the transition probabilities, and employ the Maximum Likelihood Estimation technique [Jurafsky and Martin, 1999].

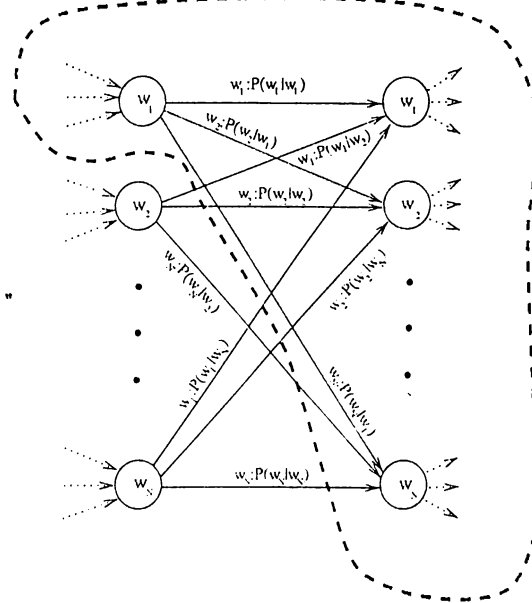
#### 2.5.4 Using HMMs for Statistical Language and Speech Processing

HMMs are useful for various language and speech processing tasks. For example, in speech recognition, if we consider each word as being generated in a state, we can use the acoustic model probabilities  $P(a_k|w_i)$  as state observation likelihoods, and the bigram language model probabilities  $P(w_i|w_{i-1})$  as the transition probabilities. We can then use the HMM algorithms to find the probability of a sequence of words given a sequence of acoustic symbols [Rabiner, 1989]. We can use HMMs for part-of-speech tagging in a similar way.

HMMs can also be used in generating parameters, i.e.  $\lambda_i$ , for deleted interpolation of  $n$ -gram language models [Jelinek, 1998]. We can construct an HMM with hidden states that enable the interpolation of multiple models. Then, the EM algorithm can be used to find the parameters, that is, the optimal weights given as probabilities for transitions entering these hidden states. Figure 2.4 is a fragment of an HMM for smoothing a bigram language model, corresponding to the marked (with a dashed line) part of the bigram language model, a fragment of which is given in Figure 2.3.

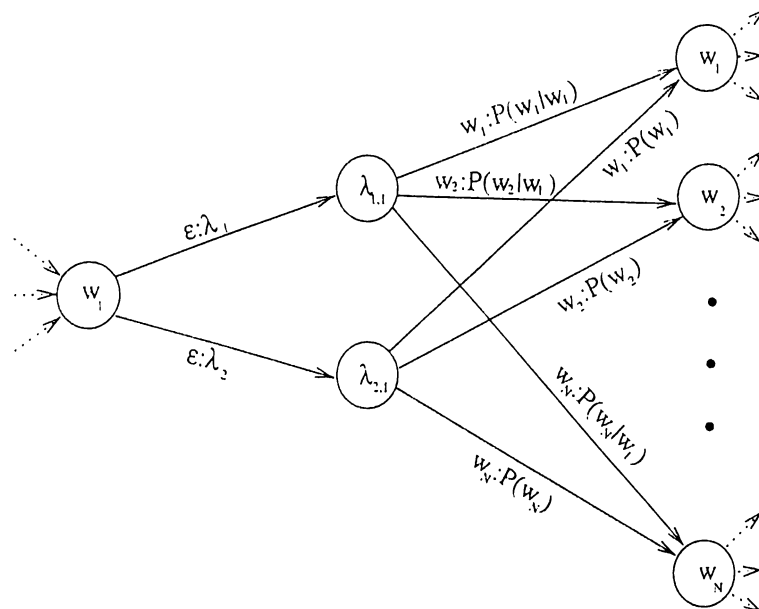
## 2.6 Maximum Entropy Models

Maximum Entropy (ME) Modeling is an approach for combining multiple information sources for classification. This approach was first proposed by Jaynes



Each transition is marked with the output produced and the probability of taking that transition.

Figure 2.3: A fragment of the Markov Model for a bigram language model.



The parameter  $\epsilon$  means that no output is generated by taking that transition.

Figure 2.4: A fragment of the HMM for estimating the parameters of a linearly interpolated bigram language model.

[1957] for statistical mechanics and has been recently successfully applied to natural language processing problems, including machine translation [Berger *et al.*, 1996], sentence boundary detection [Mikheev, 1998; Reynar and Ratnaparkhi, 1997], part-of-speech tagging [Ratnaparkhi, 1996], prepositional phrase attachment [Ratnaparkhi, 1998b], parsing [Ratnaparkhi, 1998a], statistical language modeling for speech recognition [Rosenfeld, 1996], part-of-speech tagging of inflective languages [Hajič and Iladká, 1998], and named-entity tagging [Borthwick *et al.*, 1998; Mikheev *et al.*, 1999].

### 2.6.1 The Maximum Entropy Principle

The ME approach attempts to capture all the information provided by various knowledge sources, under a single, combined model. The training data for a problem is described as a number of features, with each feature defining a constraint on the model. The ME model is the model that satisfies all the constraints with the highest entropy. The aim is selecting the most uniform model among the models that satisfy the constraints, so nothing is assumed about what is unknown. In other words, given a collection of constraints, the aim is selecting a model which is consistent with all the constraints, but otherwise as uniform as possible [Berger *et al.*, 1996].

### 2.6.2 Representing Information via Features

In the ME framework, the information is represented via features. The features  $f_i$  are binary valued functions that can be used to characterize the properties of the context  $b$  and the corresponding class  $a$ :

$$f_i : A \times B \rightarrow 0, 1$$

where  $A = \{a_1, a_2, \dots, a_n\}$  is the set of all possible classes, and  $B = \{b_1, b_2, \dots, b_k\}$  is the set of all possible contexts that we can observe. Our aim is to find an estimate for  $p(a|b)$ .

The features in this thesis are of the form:

$$f_i(a, b) = \begin{cases} 1 & \text{if } a = \bar{a} \text{ and } b \in B_i \\ 0 & \text{otherwise} \end{cases}$$

and check the co-occurrence of a class  $\bar{a}$  with an element of a set of contexts  $B_i$ , similar to those defined by Ratnaparkhi [1998a].

### 2.6.3 An Example

In order to illustrate the use of maximum entropy modeling, we are going to give a very simple example for sentence segmentation, using word categories. The task is to estimate a joint probability distribution,  $p(b, a)$ , where  $a \in A = \{0, 1\}$  and  $b \in B = \{Noun, Adjective, Verb, Preposition\}$ . The elements of  $A$  represent the presence/absence of a sentence end after the categories in  $B$ .

Suppose that we only know that:

$$p(Noun, 0) + p(Adjective, 0) + p(Verb, 0) + p(Preposition, 0) = 0.8 \quad (2.29)$$

and that:

$$\sum_{a \in A, b \in B} p(b, a) = 1. \quad (2.30)$$

The aim of our model is to predict the probability of the presence/absence of a sentence end with any category in  $B$ . We can define two features as follows:

$$f_1(a, b) = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$f_2(a, b) = 1.$$

Equations 2.29 and 2.30 are the constraints on model  $p$ 's expectations of the features:

$$E_p f_1 = 0.8 \text{ and } E_p f_2 = 1$$

where

$$E_p f_i = \sum_{a \in A, b \in B} p(a, b) \times f_i(a, b) \quad (2.31)$$

In the computation of the expectations,  $p(a, b)$  is the probability assigned by the model.

The aim of the ME framework is to maximize the entropy:

$$H(P) = - \sum_{a \in A, b \in B} p(a, b) \log p(a, b) \quad (2.32)$$

The most ‘uncertain’ way of satisfying the constraints is assigning uniform probabilities to unconstrained cases. So, the sentence segmentation model that has the maximum entropy assigns the following probabilities:

$$\begin{aligned} p_1(Noun, 0) &= 0.2 & p_1(Noun, 1) &= 0.05 \\ p_1(Adjective, 0) &= 0.2 & p_1(Adjective, 1) &= 0.05 \\ p_1(Verb, 0) &= 0.2 & p_1(Verb, 1) &= 0.05 \\ p_1(Preposition, 0) &= 0.2 & p_1(Preposition, 1) &= 0.05 \end{aligned} \quad (2.33)$$

This model is the maximum entropy model among the ones that satisfy the constraints. The entropy of this model is:

$$\begin{aligned} H(P_1) &= -(4 \times 0.2 \times \log 0.2 + 4 \times 0.05 \times \log 0.05) \\ &= 2.73 \end{aligned}$$

Another model that also satisfies the given constraints, but assumes that a non-sentence boundary is less probable with a *Preposition* and a *Verb*, than with a *Noun* or an *Adjective*.

$$\begin{aligned} p_2(Noun, 0) &= 0.3 & p_2(Noun, 1) &= 0.02 \\ p_2(Adjective, 0) &= 0.3 & p_2(Adjective, 1) &= 0.02 \\ p_2(Verb, 0) &= 0.1 & p_2(Verb, 1) &= 0.08 \\ p_2(Preposition, 0) &= 0.1 & p_2(Preposition, 1) &= 0.08 \end{aligned} \quad (2.34)$$

The entropy of the second model is:

$$H(P_2) = -(2 \times 0.3 \times \log 0.3 + 2 \times 0.1 \times \log 0.1 +$$

$$\begin{aligned}
& 2 \times 0.02 \times \log 0.02 + 2 \times 0.08 \times \log 0.08) \\
& = 2.52
\end{aligned}$$

which is lower than the first model, that assigns uniform probabilities to unconstrained cases.

### 2.6.4 Conditional Maximum Entropy Models

In this thesis, our aim will be to estimate a conditional probability distribution instead of a joint probability distribution. In previous studies using conditional maximum entropy models, the most uncertain distribution  $p^*$  that satisfies a set of  $k$  constraints is [Rosenfeld, 1996; Ratnaparkhi, 1998a]:

$$p^* = \operatorname{argmax}_{p \in P} H(p)$$

where

$$\begin{aligned}
H(p) &= - \sum_{a,b} \tilde{p}(b) p(a|b) \log p(a|b) \\
P &= \{p \mid E_p f_i = E_{\tilde{p}} f_i, i = 1, 2, \dots, k\} \\
E_{\tilde{p}} f_i &= \sum_{a,b} \tilde{p}(a, b) f_i(a, b) \\
E_p f_i &= \sum_{a,b} \tilde{p}(b) p(a|b) f_i(a, b)
\end{aligned} \tag{2.35}$$

and  $H(p)$  is the conditional entropy averaged over the training data,  $E_{\tilde{p}} f_i$  is the observed expectation of the feature  $i$  (that is, observed in the training data),  $E_p f_i$  is the model's expectation of the feature  $i$ ,  $\tilde{p}(b)$  and  $\tilde{p}(a, b)$  are the observed probabilities, and  $p(a|b)$  are the model probabilities.

### 2.6.5 Combining Information Sources

A particular way of combining evidence from multiple information sources is to weight the corresponding features in an exponential, or log-linear model:

$$p(a|b) = \frac{1}{Z(b)} \prod_{i=1}^k \alpha_i^{f_i(s,b)} \quad (2.36)$$

where  $k$  is the number of features,  $\alpha_i$  is the weight for the feature  $f_i$ , and  $Z(b)$  is a normalization constant to ensure that the resulting distribution is a valid probability distribution, and can be computed as:

$$Z(b) = \sum_a \prod_{i=1}^k \alpha_i^{f_i(s,b)} \quad (2.37)$$

Therefore, the conditional probability  $p(a|b)$  is a normalized product of the weights of the features that are 'active' on the  $(a, b)$  pair [Ratnaparkhi, 1998a]. The feature weights for the model that satisfies the Maximum Entropy Principle can be estimated using the Generalized Iterative Scaling (GIS) algorithm [Darroch and Ratcliff, 1972], which we will describe in the next section.

### 2.6.6 Parameter Estimation

The parameters of the maximum entropy model,  $p^*$ , that satisfies the set of constraints:

$$E_{p^*} f_i = E_{\tilde{p}} f_i \quad (2.38)$$

can be found using the GIS algorithm, that is guaranteed to converge to  $p^*$  [Darroch and Ratcliff, 1972]. This algorithm requires that the sum of feature values for each possible  $(a, b)$  should be equal to a constant,  $C$ :

$$\sum_{i=1}^k f_i(a, b) = C \quad (2.39)$$

If this condition is not already true, we can use the training set to choose  $C$ :

$$C = \max_{a \in A, b \in B} \sum_{i=1}^k f_i(a, b) \quad (2.40)$$



and add a correction feature  $f_{k+1}$ , such that

$$f_{k+1}(a, b) = C - \sum_{i=1}^k f_i(a, b) \quad (2.41)$$

for any  $(a, b)$  pair, as suggested by Ratnaparkhi [1998a]. In this case, unlike any other feature,  $f_{k+1}$  might get values greater than 1. A variant of the GIS algorithm, the Improved Iterative Scaling algorithm [Pietra *et al.* 1997] does not impose this constraint. But, in this thesis, we use the GIS algorithm, since adding a single correction feature is not very costly.

The GIS algorithm is as follows:

1. Compute the observed probabilities,  $\check{p}(a, b)$  and  $\check{p}(b)$ .
2. Compute the observed expectation of the features.  $E_{\check{p}}f_i$ . for each feature:

$$E_{\check{p}}f_i = \sum_{a,b} \check{p}(a, b) f_i(a, b), \quad \text{for } i = 1, 2, \dots, k$$

3. Initialize the weight,  $\alpha_i$ , for each feature:

$$\alpha_i^1 = 1, \quad \text{for } i = 1, 2, \dots, k$$

4. Compute the normalization constants,  $Z(b)$ . for each possible context:

$$Z(b) = \sum_a \prod_{i=1}^k (\alpha_i^n)^{f_i(s,b)}, \quad \forall b \in B \quad (2.42)$$

5. Compute the model probabilities:

$$p^n(a|b) = \frac{1}{Z(b)} \prod_{i=1}^k (\alpha_i^n)^{f_i(s,b)} \quad (2.43)$$

6. Compute the model's expectations of the features:

$$E_{p^n}f_i = \sum_{a,b} \check{p}(b) p^n(a|b) f_i(a, b) \quad (2.44)$$

7. Stop if

$$|E_{\bar{p}} f_i - E_{p^n} f_i| < \epsilon$$

otherwise update the model weights as follows:

$$\alpha_i^n + 1 = \alpha_i^n \left[ \frac{E_{\bar{p}} f_i}{E_{p^n} f_i} \right]^{\frac{1}{c}} \quad \text{for } i = 1, \dots, k$$

and goto step 4.\*

So, the algorithm iteratively updates the feature weights,  $\alpha_i$ , so that the model's expectation of the features becomes close to their observed expectations. Once the difference between these values is smaller than some  $\epsilon$ , the algorithm terminates. The  $\alpha$  values and the normalization constants are used to compute the probabilities that the model assigns.

# Chapter 3

## Turkish

### 3.1 Introduction

Application of statistical language modeling techniques to English (and similar languages) for natural language and speech processing tasks like parsing, word sense disambiguation, part-of-speech tagging, speech recognition, etc. has been very useful. However, languages which display a substantially different behavior than English, like Turkish, Czech, Hungarian (in that, they have agglutinative or inflective morphology and relatively free constituent order) have mainly been left unstudied. In this chapter, we will discuss the properties of Turkish, that complicate the straightforward application of traditional language modeling approaches.

### 3.2 Syntactic Properties of Turkish

#### 3.2.1 Word Order

Turkish is a free constituent order language, in which constituents at certain phrase levels can change order rather freely according to the discourse context

or text flow. The typical order of the constituents is subject-object-verb (SOV), however, other orders are also common, especially in discourse.

The morphology of Turkish enables morphological markings on the constituents to signal their grammatical roles without relying on their order. This does not mean that the word order is not important, sentences with different word orders reflect different pragmatic conditions, that is the topic, focus, and background information conveyed by those sentences differ [Erguvanlı, 1979]. For example, a constituent that is to be emphasized is generally placed immediately before the verb:<sup>1</sup>

- (1) a. Ben okula            gittim.  
       I    school+DAT go+PAST+A1SG  
       *I went to school.*
- b. Okula            ben gittim.  
       school+DAT I    go+PAST+A1SG  
       *It was me who went to school.*

Word order inside the embedded clauses is more strict; not all the variations of the order of the constituents are grammatical. A good discussion of the function of word order in Turkish grammar can be found in Erguvanlı [1979].

The variations in the word order complicates statistical language modeling, since more training data is required in order to capture the possible word order variations.

### 3.2.2 Morphology

Turkish has agglutinative morphology with productive inflectional and derivational suffixations [Oflazer, 1994]. The number of word forms one can derive from a Turkish root form may be in the millions [Hankamer, 1989]. The number

---

<sup>1</sup>DAT: dative case, PAST: past tense, A1SG: first person singular agreement.

CATEGORY	Number of Overt Morphemes		
	1	2	3
NOUN	33	490	4,825
VERB	46	895	11,313
ADJ	32	478	4,789

Table 3.1: The number of possible word formations obtained by suffixing 1, 2 and 3 morphemes to a NOUN, a VERB and an ADJECTIVE.

of possible word forms that can be obtained from a NOUN, a VERB, and an ADJECTIVE root form by suffixing 1, 2, and 3 morphemes is listed in Table 3.1. Figure 3.1 lists the 33 possible word forms that can be obtained from the noun ‘masa’ by suffixing only one morpheme.

The number of words in Turkish is theoretically infinite, since, for example, it is possible to embed multiple causatives in a single word (as in: somebody causes some other person to cause another person ... to do something). Figure 3.2 gives an example of some possible word formations from the root ‘uyu’ (‘sleep’ in English). Multiple causatives are the final examples of this table.

### 3.3 Issues for Language Modeling of Turkish

Due to the productive inflectional and derivational morphology of Turkish, the number of distinct word forms, i.e., the vocabulary size, is very large. For instance, Table 3.2 shows the size of the vocabulary for 1 and 10 million word corpora of Turkish, collected from on-line newspapers. We also give these numbers for English corpora of the same size to give an idea about the difference. This large vocabulary is the reason for a serious data sparseness problem and also significantly increases the number of parameters to be estimated even for a bigram language model. The size of the vocabulary also causes the perplexity to be large (although this is not an issue in morphological disambiguation, it is important for language modeling for speech recognition.) Table 3.3 lists the training and test set perplexities of trigram language models trained on 1 and 10 million

- 
- |               |               |
|---------------|---------------|
| 1. masaca     | 21. masayız   |
| 2. masacasına | 22. masayım   |
| 3. masacı     | 23. masada    |
| 4. masalaş    | 24. masadır   |
| 5. masalan    | 25. masan     |
| 6. masalar    | 26. masamız   |
| 7. masaları   | 27. masamsı   |
| 8. masasız    | 28. masan     |
| 9. masalı     | 29. masanız   |
| 10. masalık   | 30. masadan   |
| 11. masanın   | 31. masasal   |
| 12. masası    | 32. masasın   |
| 13. masayken  | 33. masasınız |
| 14. masaykene |               |
| 15. masayla   |               |
| 16. masaymış  |               |
| 17. masaysa   |               |
| 18. masaya    |               |
| 19. masaydı   |               |
| 20. masayı    |               |

Figure 3.1: The list of words that can be obtained by suffixing only one morpheme to a noun ‘masa’ (‘table’ in English.).

Root: uyu- ('sleep' in English)	
Some Word Formations	English Translations
uyuyorum	I am sleeping
uyuyorsun	you are sleeping
uyuyor	he/she/it is sleeping
uyuyoruz	we are sleeping
uyuyorsunuz	you are sleeping
uyuyorlar	they are sleeping
uyuduk	we slept
uyudukça	as long as (somebody) sleeps
uyumalıyız	we must sleep
uyumadan	without sleeping
uyuman	your sleeping
uyurken	while (somebody) is sleeping
uyuyunca	when (somebody) sleeps
uyutmak	to cause somebody to sleep
uyutturmak	to cause (somebody) to cause (another person) to sleep
uyutturtturmak	to cause (somebody) to cause (some other person) to cause (another person) to sleep

Figure 3.2: Example of possible word formations with derivational and inflectional suffixes from a Turkish verb.

Language	Corpus size	Vocabulary size
Turkish	1M words	106,547
	10M words	417,775
English	1M words	33,398
	10M words	97,734

Table 3.2: Vocabulary sizes for two Turkish and English corpora.

word corpora for Turkish and English. For each corpus, the first column is the perplexity for the data the language model is trained on, and the second column is the perplexity for previously unseen test data of 1 million words. Note that the trigram-language model perplexity that we found for English is very close to the one reported by Brown et.al [1992a].

Language	Training Data	Training Set Perplexity	Test Set (1M words) Perplexity
Turkish	1M words	66.13	1449.81
	10M words	94.08	1084.13
English	1M words	43.29	161.16
	10M words	44.38	108.52

Table 3.3: The perplexity of Turkish and English corpora using word-based trigram language models.

Another major reason for the high perplexity of Turkish is the high percentage of out-of-vocabulary words (words in the test data which do not occur in the training data): this also results from the productivity of the word formation process.

The issue of large vocabulary brought in by productive inflectional and derivational morphology also makes tagset design an important issue. In languages like English, the number of POS tags that can be assigned to the words in a text is rather limited (less than 100) (though some researchers have used large tag sets to refine granularity, but they are still small compared to Turkish.)<sup>2</sup> But, such a finite tagset approach for languages like Turkish may lead to an inevitable loss of

<sup>2</sup>More information on the issues of tagset design is given by Elworthy [1995].



information. The reason for this is that the morphological features of intermediate derivations can contain markers for syntactic relationships. Thus, leaving out this information within a fixed-tagset scheme may prevent crucial syntactic information from being represented [Oflazer *et al.*, 1999]. For example, it is not clear what POS tag should be assigned to the word *sağlamlaştırarak*, without losing any information: the category of the root (Adjective), the final category of the word as a whole (Noun) or one of the intermediate categories (Verb).<sup>3</sup> Ignoring the fact that the root word is an adjective may sever any relationships with an adverbial modifier modifying the root.

*sağlam+laş+tır+mak*  
 sağlam+Adj<sup>DB</sup>+Verb+Become<sup>DB</sup>  
 +Verb+Caus+Pos<sup>DB</sup>+Noun+Inf+A3sg+Pnon+Nom  
 to cause (something) to become strong /  
 to strengthen (something)

Thus instead of a simple POS tag, we use the full morphological analyses of the words, represented as a combination of features (including any derivational markers) as their morphosyntactic tags. For instance in the example above, we would use everything including the root form as the morphosyntactic tag.

### 3.4 Examples of Morphological Ambiguity

In this section, we will give an example of morphological ambiguity in Turkish, using the word ‘izin’, which occurs in the three sentences below:

1. Yerdeki izin temizlenmesi gerek.

*The trace on the floor should be cleaned.*

---

<sup>3</sup>The morphological features other than the POSs are: +Become: become verb, +Caus: causative verb, +Pos: Positive polarity, +Inf: marker that derives an infinitive form from a verb, +A3sg: 3sg number-person agreement, +Pnon: No possessive agreement, and +Nom: Nominative case. <sup>DB</sup>’s mark derivational boundaries.

2. Üzerinde parmak izin kalmış.  
*Your finger print is left on (it).*
3. İçeri girmek için izin alman gerekiyor.  
*You need a permission to enter.*

and the following are the corresponding morphological analysis, respectively:

izin

1. iz+Noun+A3sg+Pnon+Gen (trace/print)
2. iz+Noun+A3sg+P2sg+Nom (trace/print)
3. izin+Noun+A3sg+Pnon+Nom (permission)

Further examples of morphological ambiguity, and a classification of frequent types of ambiguities is given in the M.Sc. thesis of Tür [1996].

### 3.5 Inflectional Groups

In order to alleviate the data sparseness problem we break down the full tags into smaller units. We represent each word as a sequence of *inflectional groups* (IGs hereafter), separated by  $\wedge$ DBs denoting derivation boundaries, as described by Oflazer [1999]. Thus a morphological parse is represented in the following general form:

$$\text{root} + \text{IG}_1 \wedge \text{DB} + \text{IG}_2 \wedge \text{DB} + \dots \wedge \text{DB} + \text{IG}_n$$

where  $\text{IG}_i$  denotes relevant inflectional features of the inflectional groups, including the part-of-speech for the root or any of the derived forms.

For example, the infinitive form *sağlamlaştırmak* given in Section 3.3 would be represented with the adjective reading of the root *sağlam* and the following 4 IGs:

1. Adj
2. Verb+Become
3. Verb+Caus+Pos
4. Noun+Inf+A3sg+Pnon+Nom

In order to simplify our models further, we will use the following property of the dependency grammar for Turkish: When a word is considered as a sequence of IGs, syntactic relation links only emanate from the last IG of a (dependent) word, and land on one of the IGs of the (head) word on the right, as shown in Figure 3.3 [Oflazer, 1999]. Figure 3.4 shows an example sentence with the dependency relations marked, taken from [Oflazer, 1999]. In this example, the words are segmented along the IG boundaries, marked with a '+' sign. The inflectional suffixes are marked preceding '-' sign.

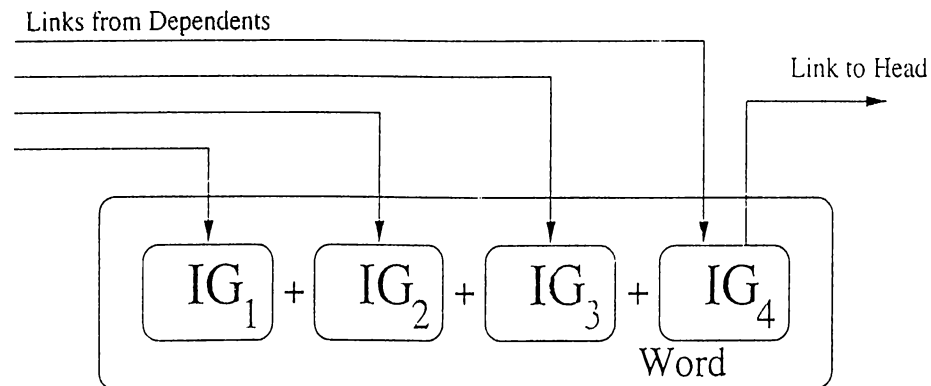


Figure 3.3: Inflectional groups in a word and the syntactic relation links.

### 3.6 Statistics on the Inflectional Groups

The number of possible units to be modeled is important for statistical language modeling. Table 3.4 provides a comparison of the number distinct full morphosyntactic tags (ignoring the root words in this case) and IGs, generatively possible and observed in a corpus of 1M words (considering all ambiguities). As we already

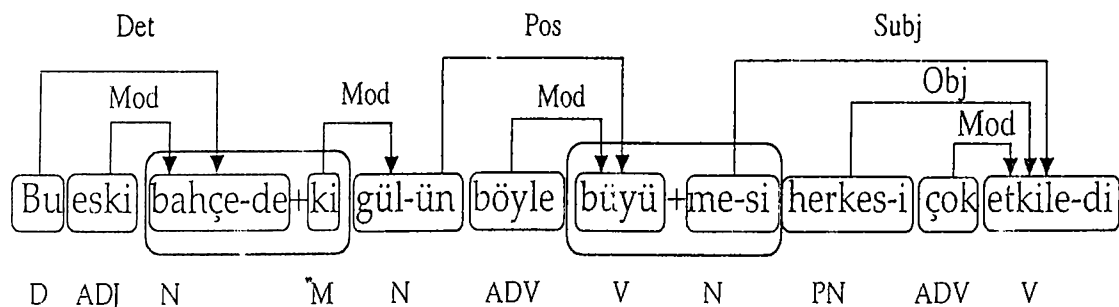


Figure 3.4: An example dependency tree for a Turkish sentence. The words are segmented along the IG boundaries.

	Possible	Observed
Full Tags (No roots)	$\infty$	10,531
Inflectional Groups	9,129	2,194

Table 3.4: Numbers of Tags and IGs

mentioned, the number of possible tags that are theoretically possible is infinite for Turkish. In a 1 million word corpus, collected from Turkish daily newspapers, we observed 10,531 part-of-speech tags ignoring the root words, which is very high. The number of possible IGs is 9,129 and we observed 2,194 IGs in our corpus. The number of observed IGs is still very high, but smaller than the number of full tags (as expected).

On the average, in running text of about 850K tokens, there are 1.76 morphological parses/token and 1.38 IGs/parse. 55% of the tokens have only one parse. Of all the parses:

- 72% have 1 IG,
- 18% have 2 IGs.
- 7% have 3 IGs,
- 2% have 4 IGs, and
- 1% have 5 or more IGs.

There are also parses which have 7 or 8 IGs. However, these are very rare, as can be seen from the statistics. Since most of the parses have only one or two inflectional groups, dividing morphological parses from their derivational boundaries, should not complicate the processing for our tasks.

•

# Chapter 4

## Related Work

### 4.1 Introduction

In order to enhance the presentation of our motivation, we will review some of the previous studies on part-of-speech tagging, morphological disambiguation, statistical language modeling, and language and speech processing systems for Turkish and other similar, morphologically rich languages, in this chapter.

### 4.2 Part-of-Speech Tagging

There has been a large number of studies in tagging and morphological disambiguation using various techniques. Part-of-speech tagging systems have used either a rule-based or a statistical approach.

In the rule-based approach, first, a dictionary is used to assign each word a list of potential part-of-speech tags, then a large number of hand-crafted linguistic constraints are used to eliminate impossible tags or morphological parses for a given word in a given context [Karlsson *et al.*, 1995].

In the statistical approach, a large, labeled corpus has been used to train a

probabilistic model which then has been used to tag new text, assigning the most likely tag for a given word in a given context (e.g., Church [1988], Garside [1988], DeRose [1988]). It is also possible to train a stochastic tagger using unlabeled data, with the expectation maximization algorithm (e.g. Cutting et al. [1992]).

Part-of-Speech tagging is one of the first NLP tasks, for which Maximum Entropy Modeling approach has been used [Ratnaparkhi. 1996].

Brill [1995a] has presented a transformation-based learning approach, which induces disambiguation rules from tagged corpora.

Morphological disambiguation in inflecting or agglutinative languages with complex morphology involves more than determining the major or minor parts-of-speech of the lexical items. Typically, morphology marks a number of inflectional or derivational features and this involves ambiguity. For instance, a given word may be chopped up in different ways into morphemes, a given morpheme may mark different features depending on the morphotactics, or lexicalized variants of derived words may interact with productively derived versions. We assume that *all syntactically relevant features* of word forms have to be determined correctly for morphological disambiguation.

In this context, there have been some interesting previous studies for different languages. Levinger et al. [1995] have reported on an approach that learns morpholexical probabilities from an untagged corpus and have used the resulting information in morphological disambiguation of Hebrew. Hajič and Hladká [1998] have used maximum entropy modeling approach for morphological disambiguation of Czech, which is an inflecting language. Ezeiza et al. [1998] have combined stochastic and rule-based disambiguation methods for Basque, which is also an agglutinative language. Megyesi [1999] has adapted Brill's POS tagger with extended lexical templates to Hungarian.

Previous approaches to morphological disambiguation of Turkish text had employed a constraint-based approach [Oflazer and Kuruöz, 1994; Oflazer and Tür. 1996; Oflazer and Tür, 1997]. Although results obtained earlier in these approaches were reasonable, the fact that the constraint rules were hand crafted

posed a rather serious impediment to the generality and improvement of these systems.

### 4.3 Statistical Language Modeling

Statistical language modeling roots back to Markov's work on predicting whether an upcoming letter in Pushkin's *Eugene Onegin* would be a vowel or a consonant, using bigrams and trigrams [Jurafsky and Martin, 1999]. Shannon used  $n$ -grams to compute the entropy of English and to compute approximations to English word sequences [Shannon, 1948].

Today, statistical language modeling techniques are used successfully in almost all natural language and speech processing tasks. For more information about statistical language modeling techniques and the application of these techniques to language processing tasks, the reader is referred to textbooks on the subject (some recent good examples are, Manning and Schütze [1999] and Jurafsky and Martin [1999]).

Before proceeding to related work for Turkish, we would like to mention a study for large vocabulary continuous speech recognition (LVCSR) of Korean, which is an agglutinative language. Unlike the previous approaches based on the sequence words or letters, this study suggests the use of syllables as language modeling units, to overcome the problem of large vocabulary size and so to reduce perplexity [Kiecza *et al.*, 1999]. However, due to the shortness of syllables, their acoustic confusability is high (which is important for speech recognition) and a standard trigram language model using syllables has very limited scope (which is also a problem for our case). So, a unit that lies between the two extremes, syllables and words, should be used. In that study, a data-driven approach is used to find the appropriate unit for language modeling.

Another study, again on Korean compares the performance of LVCSR systems using syllables and morphemes as the recognition unit [Kwon, 2000]. With the syllables, the percentage of out-of-vocabulary is nearly zero, but they also note



that syllables have high acoustic confusability. Their best results are obtained using units that are formed merging morphemes.

A recent study which also uses morphemes for language modeling of English reports that lower word error rates can be obtained in speech recognition by decomposing words into their component morphemes [Fang and Huckvale, 2000]. They use phonologically constrained morphological analysis (PCMA) that divides words into their morphemes conditioning both on their orthography and pronunciation. The benefits of PCMA include: reduced lexicon size, reduced perplexity, reduced language model size, and reduced word error rate.

## 4.4 Turkish

Almost all of the systems developed for processing Turkish text until today, can be considered as rule-based. We can list the systems developed for Turkish as follows:

- a morphological analyzer using finite state transducers [Oflazer, 1994],
- a morphological disambiguator using voting constraints [Oflazer and Tür, 1997],
- a syntactic parser using the Lexical Functional Grammar formalism [Güngördü and Oflazer, 1995],
- a dependency parser using an extended finite state approach [Oflazer, 1999],
- a parser using the government binding approach [Birtürk, 1998],
- a parser using sign-based phrase structure grammars [Şehitoğlu, 1996],
- spelling checkers [Solak and Oflazer, 1993] and correctors [Oflazer and Güzey, 1994; Oflazer, 1996].

- two machine translation systems from English to Turkish, one using structural mapping [Çiğdem Keyder Turhan, 1997] and the other using interlingua-based methods [Hakkani *et al.*, 1998],
- a large vocabulary isolated word speech recognition system [Yılmaz, 1999], and
- a large vocabulary continuous speech recognition system for agglutinative languages, including Turkish [Çarkı *et al.*, 2000].

We believe that this thesis is the first work examining statistical language modeling techniques for understanding Turkish, together with a concurrent Ph.D. thesis on information extraction from Turkish using statistical techniques [Tür, 2000].

# Chapter 5

## Morphological Disambiguation

### 5.1 Introduction

Morphological disambiguation is the problem of selecting the sequence of morphological parses (including the root).  $T = t_1^n = t_1, t_2, \dots, t_n$ , corresponding to a sequence of words  $W = w_1^n = w_1, w_2, \dots, w_n$ , from the set of possible parses for these words.

For example, the words of the Turkish noun phrase have the parses given below:

*evin*

1. **evin+Noun+A3sg+Pnon+Nom**
2. ev+Noun+A3sg+P2sg+Nom
3. **ev+Noun+A3sg+Pnon+Gen**

*terasi*

1. **teras+Noun+A3sg+P3sg+Nom**
2. teras+Noun+A3sg+Pnon+Acc

The correct parse for each word is given in boldface. Among the possible parse combinations, only the first parse of the first word with the first parse of the second word, and the third parse of the first word with the first parse of the second

word make up a grammatical noun phrase. Among the grammatical combinations, only the root of the third parse of the first word occurs frequently with the root of the first parse of the second word.

Morphological Disambiguation is a useful prior step for syntactic parsing, since it decreases the ambiguity of the sentence, and hence makes the computational problem smaller [Voutilainen, 1998]. Spelling correction and text to speech synthesis systems can also benefit from a morphological disambiguator for context sensitive selection of correct pronunciation and for selection of true spellings, respectively.

Our approach is to model the distribution of morphological parses given the words, using a hidden Markov model, and then to seek the variable  $T$ , that maximizes  $P(T|W)$ :

$$\operatorname{argmax}_T P(T|W) = \operatorname{argmax}_T \frac{P(T) \times P(W|T)}{P(W)} \quad (5.1)$$

$$= \operatorname{argmax}_T P(T) \times P(W|T) \quad (5.2)$$

The term  $P(W)$  is a constant for all choices of  $T$ , and can thus be ignored when choosing the most probable  $T$ . Thus, Equation 5.1 can be simplified into Equation 5.2.

We can simplify the problem of morphological disambiguation using following assumptions [Manning and Schütze, 1999]:

- Words are independent of each other, given their tags, that is,

$$P(W|T) = \prod_{i=1}^n P(w_i|t_i^n),$$

and,

- A word's identity depends only on its tag, and not on previous words or tags, that is,

$$P(w_i|t_1^n) = P(w_i|t_i).$$

We can then compute  $P(W|T)$  as follows:

$$P(W|T) = \prod_{i=1}^n P(w_i|t_i^n) = \prod_{i=1}^n P(w_i|t_i) \quad (5.3)$$

We can compute  $P(T)$  using the chain rule:

$$P(T) = \prod_{i=1}^n P(t_i|t_1^{i-1}) \quad (5.4)$$

We can simplify Equation 5.4 further with the trigram tag model, so:

$$P(T) = \prod_{i=1}^n P(t_i|t_{i-2}, t_{i-1}) \quad (5.5)$$

Therefore, equation can be computed as:

$$\operatorname{argmax}_T P(T|W) = \operatorname{argmax}_T \prod_{i=1}^n P(t_i|t_{i-2}, t_{i-1}) \times P(w_i|t_i) \quad (5.6)$$

This is the basic formulation of part-of-speech tagging for languages like English [Charniak *et al.*, 1993; Merialdo, 1994; Dermatas and Kokkinakis, 1995; Brants, 2000], and also is the basis of our baseline model where we use the full morphological analysis including the root word as the tag of the word. In the remaining of this chapter, we will use the terms tag, morphological analysis or parse interchangeably, to refer to individual distinct morphological parses of a token.

## 5.2 Simplifying the problem for Turkish

In Turkish, given a morphological analysis including the root, there is only one surface form that can correspond to it, that is, there is no morphological generation ambiguity.<sup>1</sup> Therefore, we can assume that  $P(w_i|t_i) = 1$  in the formulation

---

<sup>1</sup>This is almost always true. There are a few word forms like *gelirkene* and *nerde*, which have the same morphological parses with the word forms *gelirken* and *nerede*, respectively but are pronounced (and written) slightly differently. These are very rarely seen in written texts, and can thus be ignored.

above, since  $t_i$  includes the root form and all morphosyntactic features to uniquely determine the word form. In our case,

$$P(w_i|t_1^n) = P(w_i|t_i) = 1, \quad (5.7)$$

therefore, we can write:

$$P(W|T) = \prod_{i=1}^n P(w_i|t_i^n) = 1 \quad (5.8)$$

and the morphological disambiguation problem can be represented as below:

$$\operatorname{argmax}_T P(T|W) = \operatorname{argmax}_T P(T) \quad (5.9)$$

### 5.3 Morphological Disambiguation of Turkish with $n$ -gram Language Models

We use trigram language models for morphologically disambiguating Turkish words. The probability of a sequence of tags,  $P(T)$  can be computed as follows according to the chain rule:

$$P(T) = P(t_n|t_1^{n-1}) \times P(t_{n-1}|t_1^{n-2}) \times \dots \times P(t_2|t_1) \times P(t_1) \quad (5.10)$$

Simplifying Equation 5.10 further with a trigram tag model, we get:

$$\begin{aligned} P(T) &= P(t_n|t_{n-2}, t_{n-1}) \times P(t_{n-1}|t_{n-3}, t_{n-2}) \times \dots \times P(t_3|t_1, t_2) \times P(t_2|t_1) \times P(t_1) \\ &= \prod_{i=1}^n P(t_i|t_{i-2}, t_{i-1}) \end{aligned} \quad (5.11)$$

where we define  $P(t_1|t_{-1}, t_0) = P(t_1)$ ,  $P(t_2|t_0, t_1) = P(t_2|t_1)$  to simplify the notation.

In order to determine the trigram probabilities, we use the Maximum Likelihood Estimation technique, and smooth the probabilities using Good-Turing method [Gale, 1994] combined with the Backoff modeling [Katz, 1987].

The tag sequence that we are looking for, is the tag sequence that has the maximum probability according to our trigram tag model (see equation 5.9).

### 5.3.1 Using IGs for Morphological Disambiguation

If we consider morphological analyses as a sequence of root and IGs, each parse  $t_i$  can be represented as  $(r_i, IG_{i,1}, \dots, IG_{i,n_i})$ , where  $n_i$  is the number of IGs in the  $i^{th}$  word [Hakkani-Tür *et al.*, 2000].<sup>2</sup>  $r_i$  includes the part-of-speech of the root word. This representation changes the problem as follows:

$$\begin{aligned} P(t_i|t_1^{i-1}) &= P(t_i|t_{i-2}, t_{i-1}) \\ &= P((r_i, IG_{i,1} \dots IG_{i,n_i})|(r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}, \\ &\quad (r_{i-1}, IG_{i-1,1} \dots IG_{i-1,n_{i-1}})) \end{aligned} \quad (5.12)$$

We can use the chain rule to factor out the individual components:

$$\begin{aligned} P(t_i|t_1^{i-1}) &= P(r_i|(r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}, (r_{i-1}, IG_{i-1,1} \dots IG_{i-1,n_{i-1}})) \times \\ &\quad P(IG_{i,1}|(r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}, (r_{i-1}, IG_{i-1,1} \dots IG_{i-1,n_{i-1}}), r_i) \times \\ &\quad \dots \times \\ &\quad P(IG_{i,n_i}|(r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}, \\ &\quad (r_{i-1}, IG_{i-1,1} \dots IG_{i-1,n_{i-1}}), r_i, IG_{i,1}, \dots, IG_{i,n_i-1})) \end{aligned} \quad (5.13)$$

This formulation still suffers from the data sparseness problem, since the parameter space is still very large. To alleviate this, we make the following simplifying assumptions:

1. A root word depends only on the roots of the previous words, and is independent of the inflectional and derivational productions on them:

$$\begin{aligned} P(r_i|(r_{i-2}, IG_{i-2,1}, \dots, IG_{i-2,n_{i-2}}), \\ (r_{i-1}, IG_{i-1,1}, \dots, IG_{i-1,n_{i-1}})) &= P(r_i|r_{i-2}, r_{i-1}) \end{aligned} \quad (5.14)$$

---

<sup>2</sup>In our training and test data, the number of IGs in a word form is on the average 1.6, therefore,  $n_i$  is usually 1 or 2. We have seen, occasionally, word forms with 5 or 6 inflectional groups.

The intention here is that this will be useful in the disambiguation of the root word when a given form has morphological parses with different root words. So, for instance, for disambiguating the surface form *adam* with the following two parses:

- (a) adam+Noun+A3sg+Pnon+Nom (*man*)
- (b) ada+Noun+A3sg+P1sg+Nom (*my island*)

in the noun phrase *kırmızı kazaklı adam* (*the man with a red sweater*), only the roots (along with the part-of-speech of the root) of the previous words will be used to select the right root.

Note that the selection of the root has some impact on what the next IG in the word is, but it is very hard to isolate this, so we assume that IGs are determined by the syntactic context and not by the root.

2. An interesting observation that we can make about Turkish is that when a word is considered as a sequence of IGs, syntactic relations are only between the last IG of a (dependent) word and with some (including the last) IG of the (head) word on the right [Oflazer, 1999].<sup>3</sup>

Based on these assumptions, we define three models, all of which are based on word level trigrams. In the following subsections, we will describe each of these models.

### Model 1

In Model 1, the presence of IGs in a word only depends on the final IGs of the last two words. That is, in order to estimate the probability of an IG in a word, we only look at the final IGs of the previous two words, as shown in Figure 5.1. This model ignores any morphotactical relation between an IG and any previous IG in the same word. Therefore, the probability of an IG is estimated as follows:

---

<sup>3</sup>There are minor exceptions to this, especially in conversations. With some word orderings, the dependency relation links might land on an IG of the word on the left.



**Model 1:** The root of a word depends only on the roots of the previous two words, and an IG of a word depends only on the final IGs of the previous two words in our first model, as shown with underlining and boldfacing.

$$\begin{array}{l}
 t_{i-2} : \underline{r_{i-2}} \quad \underline{IG_{i-2,1}} \quad IG_{i-2,2} \quad \dots \quad IG_{i-2,n_{i-2}-1} \quad \mathbf{IG_{i-2,n_{i-2}}} \\
 t_{i-1} : \underline{r_{i-1}} \quad \underline{IG_{i-1,1}} \quad IG_{i-1,2} \quad \dots \quad IG_{i-1,n_{i-1}-1} \quad \mathbf{IG_{i-1,n_{i-1}}} \\
 t_i : \underline{r_i} \quad \underline{IG_{i,1}} \quad IG_{i,2} \quad \dots \quad IG_{i,k-1} \quad \mathbf{IG_{i,k}} \quad IG_{i,k+1} \quad \dots \quad IG_{i,n_i}
 \end{array}$$

Figure 5.1: The dependency between current word and its history word according to Model 1.

$$\begin{aligned}
 P(IG_{i,k} | (r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}, \\
 (r_{i-1}, IG_{i-1,1}, \dots, IG_{i-1,n_{i-1}}), r_i, IG_{i,1}, \dots, IG_{i,k-1}) = \\
 P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}})
 \end{aligned} \tag{5.15}$$

As a result, the probability of an analysis given the previous two analyses is estimated as follows:

$$\begin{aligned}
 P(t_i | t_{i-2}, t_{i-1}) = P(r_i | r_{i-2}, r_{i-1}) \times \\
 \prod_{k=1}^{n_i} P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}})
 \end{aligned} \tag{5.16}$$

The first factor is the relationship between the roots, as shown with underlining in Figure 5.1, the second factor (which itself is the product of probabilities) models the relationship between the IGs, as shown with boldfacing in Figure 5.1.

## Model 2

In Model 2, we use the assumption that the presence of IGs in a word only depends on the final IGs of the previous two words and the previous IG in the same word, as shown in Figure 5.2. In this model, we consider morphotactical relations and assume that an IG (except the first one) in a word form has some dependency on previous IGs. Given that on the average a word has about 1.6 IGs, IG bigrams should be sufficient. Therefore, the probability of an IG is estimated as follows:

$$P(IG_{i,k} | (r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}),$$

**Models 2 and 3:** The root of a word depends only on the roots of the previous two words, and an IG of a word depends only on the final IGs of the previous two words and the previous IG in the same word. In our second model, as shown with underlining and boldfacing.

$$\begin{array}{l}
 t_{i-2} : \underline{r_{i-2}} \quad \underline{IG_{i-2,1}} \quad IG_{i-2,2} \quad \dots \quad IG_{i-2,n_{i-2}-1} \quad \mathbf{IG_{i-2,n_{i-2}}} \\
 t_{i-1} : \underline{r_{i-1}} \quad \underline{IG_{i-1,1}} \quad IG_{i-1,2} \quad \dots \quad IG_{i-1,n_{i-1}-1} \quad \mathbf{IG_{i-1,n_{i-1}}} \\
 t_i : \quad \underline{r_i} \quad \underline{IG_{i,1}} \quad IG_{i,2} \quad \dots \quad \mathbf{IG_{i,k-1}} \quad \mathbf{IG_{i,k}} \quad IG_{i,k+1} \quad \dots \quad IG_{i,n_i}
 \end{array}$$

Figure 5.2: The dependency between current word and its history word according to Models 2 and 3.

$$\begin{aligned}
 & (r_{i-1}, IG_{i-1,1}, \dots, IG_{i-1,n_{i-1}}) \cdot r_i \cdot IG_{i,1}, \dots, IG_{i,k-1} = \\
 & P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, \mathbf{IG_{i,k-1}})
 \end{aligned}$$

Therefore, we compute the probability of a morphological analysis given the previous two analyses as follows:

$$\begin{aligned}
 P(t_i | t_{i-2}, t_{i-1}) &= P(r_i | r_{i-2}, r_{i-1}) \\
 &\times \prod_{k=1}^{n_i} P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, \mathbf{IG_{i,k-1}}) \quad (5.17)
 \end{aligned}$$

### Model 3

Model 3 uses the same assumptions with Model 2, except that the dependence on the previous IG in a word is assumed to be independent of the dependence on the final IGs of the previous words. This allows the formulation to separate the contributions of the morphotactics and local syntax. That is,

$$\begin{aligned}
 & P(IG_{i,k} | (r_{i-2}, IG_{i-2,1} \dots IG_{i-2,n_{i-2}}), \\
 & \quad (r_{i-1}, IG_{i-1,1}, \dots, IG_{i-1,n_{i-1}}) \cdot r_i \cdot IG_{i,1}, \dots, IG_{i,k-1}) = \\
 & P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,k-1})
 \end{aligned}$$

and,

$$\begin{aligned}
 & P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,k-1}) \\
 &= \frac{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,k-1} | IG_{i,k}) \times P(IG_{i,k})}{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,k-1})}
 \end{aligned}$$

$$\begin{aligned}
&= \frac{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}} | IG_{i,k}) \times P(IG_{i,k-1} | IG_{i,k}) \times P(IG_{i,k})}{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}) \times P(IG_{i,k-1})} \\
&= \frac{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}} | IG_{i,k}) \times P(IG_{i,k})}{P(IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}})} \times \\
&\quad \frac{P(IG_{i,k-1} | IG_{i,k}) \times P(IG_{i,k})}{P(IG_{i,k-1})} \times \frac{1}{P(IG_{i,k})} \\
&= \frac{P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}) \times P(IG_{i,k} | IG_{i,k-1})}{P(IG_{i,k})} \tag{5.18}
\end{aligned}$$

Therefore,

$$\begin{aligned}
P(t_i | t_{i-2}, t_{i-1}) &= P(r_i | r_{i-2}, r_{i-1}) \\
&\quad \times \prod_{k=1}^{n_i} (P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}) \times \\
&\quad \quad \quad \frac{P(IG_{i,k} | IG_{i,k-1})}{P(IG_{i,k})}) \tag{5.19}
\end{aligned}$$

In order to simplify the notation in the description of our models, we have defined the following:

$$\begin{aligned}
P(r_1 | r_{-1}, r_0) &= P(r_1) \\
P(r_2 | r_0, r_1) &= P(r_2 | r_1) \\
P(IG_{1,k} | IG_{-1,n_{-1}}, IG_{0,n_0}) &= P(IG_{1,k}) \\
P(IG_{2,l} | IG_{0,n_0}, IG_{1,n_1}) &= P(IG_{2,l} | IG_{1,n_1}) \\
P(IG_{i,1} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,0}) &= P(IG_{i,1} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}) \\
P(IG_{1,k} | IG_{-1,n_{-1}}, IG_{0,n_0}, IG_{1,k-1}) &= P(IG_{1,k} | IG_{1,k-1}) \\
P(IG_{2,l} | IG_{0,n_0}, IG_{1,n_1}, IG_{2,l-1}) &= P(IG_{2,l} | IG_{1,n_1}, IG_{2,l-1}) \\
P(IG_{2,1} | IG_{1,n_1}, IG_{2,0}) &= P(IG_{2,1} | IG_{1,n_1}) \\
P(IG_{i,1} | IG_{i,0}) &= P(IG_{i,1})
\end{aligned}$$

for  $k = 1, 2, \dots, n_1$ ,  $l = 1, 2, \dots, n_2$ , and  $i = 1, 2, \dots, n$ .

We also have built a baseline model based on the standard definition of the tagging problem in Equation 5.1. For the baseline, we have assumed that the part of the morphological analysis after the root word is the tag in the conventional sense (and the assumption that  $P(w_i | t_i) = 1$  no longer holds).

### 5.3.2 An Example

In this section, we will give an example of how our three models compute the probability of the sequence of analyses, given their surface forms. For example, the probability of the analysis sequence:

kırmızı+Adj  
 kazak+Noun+A3sg+Pnon+Nom^DB+Adj+With  
 adam+Noun+A3sg+Pnon+Nom

given the surface form 'kırmızı kazaklı adam' ('the man with a red sweater' in English) is computed as follows:

According to Model 1,

$$P(\text{kırmızı} + \text{Adj}, \text{kazak} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}^{\text{DB}} + \text{Adj} + \text{With},$$

$$\text{adam} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | \text{kırmızı}, \text{kazaklı}, \text{adam}) =$$

$$P(r_1 = \text{kırmızı} + \text{Adj}) \times$$

$$P(r_2 = \text{kazak} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}) \times$$

$$P(r_3 = \text{adam} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}, r_2 = \text{kazak} + \text{Noun}) \times$$

$$P(IG_{1,1} = \text{Adj}) \times$$

$$P(IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}) \times$$

$$P(IG_{2,2} = \text{Adj} + \text{With} | IG_{1,1} = \text{Adj}) \times$$

$$P(IG_{3,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}, IG_{2,2} = \text{Adj} + \text{With})$$

According to Model 2,

$$P(\text{kırmızı} + \text{Adj}, \text{kazak} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}^{\text{DB}} + \text{Adj} + \text{With},$$

$$\text{adam} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | \text{kırmızı}, \text{kazaklı}, \text{adam}) =$$

$$P(r_1 = \text{kırmızı} + \text{Adj}) \times$$

$$P(r_2 = \text{kazak} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}) \times$$

$$\begin{aligned}
& P(r_3 = \text{adam} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}, r_2 = \text{kazak} + \text{Noun}) \times \\
& P(IG_{1,1} = \text{Adj} | IG_{1,0} = \langle \text{ROOT} \rangle) \times \\
& P(IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}, \\
& \quad IG_{2,0} = \langle \text{ROOT} \rangle) \times \\
& P(IG_{2,2} = \text{Adj} + \text{With} | IG_{1,1} = \text{Adj}, \\
& \quad IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}) \times \\
& P(IG_{3,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}, \\
& \quad IG_{2,2} = \text{Adj} + \text{With}, IG_{3,0} = \langle \text{ROOT} \rangle)
\end{aligned}$$

Note that, we use the symbol  $\langle \text{ROOT} \rangle$ , when estimating the probability of each word's initial IG to emphasize that there is no IG between the current IG and the root, that is, the current IG is the first IG of the corresponding word form.

According to Model 3,

$$\begin{aligned}
& P(\text{kırmızı} + \text{Adj}, \text{kazak} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} \text{DB} + \text{Adj} + \text{With}, \\
& \quad \text{adam} + \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | \text{kırmızı}, \text{kazaklı}, \text{adam}) \\
& P(r_1 = \text{kırmızı} + \text{Adj}) \times \\
& P(r_2 = \text{kazak} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}) \times \\
& P(r_3 = \text{adam} + \text{Noun} | r_1 = \text{kırmızı} + \text{Adj}, r_2 = \text{kazak} + \text{Noun}) \times \\
& P(IG_{1,1} = \text{Adj} | IG_{1,0} = \langle \text{ROOT} \rangle) \times \\
& P(IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}) \times \\
& \frac{P(IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{2,0} = \langle \text{ROOT} \rangle)}{P(IG_{2,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom})} \times \\
& P(IG_{2,2} = \text{Adj} + \text{With} | IG_{1,1} = \text{Adj}) \times \\
& \frac{P(IG_{2,2} = \text{Adj} + \text{With} | IG_{2,0} = \langle \text{ROOT} \rangle)}{P(IG_{2,2} = \text{Adj} + \text{With})} \times \\
& P(IG_{3,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{1,1} = \text{Adj}, \\
& \quad IG_{2,2} = \text{Adj} + \text{With}) \times \\
& \frac{P(IG_{3,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} | IG_{3,0} = \langle \text{ROOT} \rangle)}{P(IG_{3,1} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom})}
\end{aligned}$$

### 5.3.3 Implementation of the Models

The models described above require two types of probabilities for the computation of the probabilities of the correct morphological analysis: root probabilities and IG probabilities. One way to construct the models is to form the root and IG models that give us an estimate for the root and IG trigram probabilities, and then merge these two models by computing the probabilities of all possible morphological analysis sequences. However, the number of these sequences is infinite, because of the derivational morphology, so it is impossible to construct the complete model, that has the probabilities for all possible trigram sequences.

However, it is possible to construct a model for the test data at run-time, by taking the product of the root and IG probabilities. The model will not be complete, but we will only compute the probabilities for the sequences that we will need as we try to find the most probable tag sequence. Figure 5.3 shows the sequence of steps for combining the two models.

We first count the root and IG sequences in the training data. Using these counts, and the SRILM – the SRI language modeling toolkit [Stolcke, 1999], we form two trigram models that estimate the root and IG probabilities. SRILM is a toolkit for building and applying statistical language models (LMs), primarily for use in speech recognition, statistical tagging and segmentation. The toolkit builds  $n$ -gram language models in ARPA  $n$ -gram format.

We construct the combined models using the test data and the root and IG models, at run-time, and use the Viterbi Algorithm to find the most probable tag sequence. Figure 5.4 shows an example hidden Markov model we use for morphological disambiguation of each sentence in our test data. The state output probabilities of this HMM are set to one, and we use the trigram and bigram language model probabilities as state transition probabilities, which are computed according to our models.

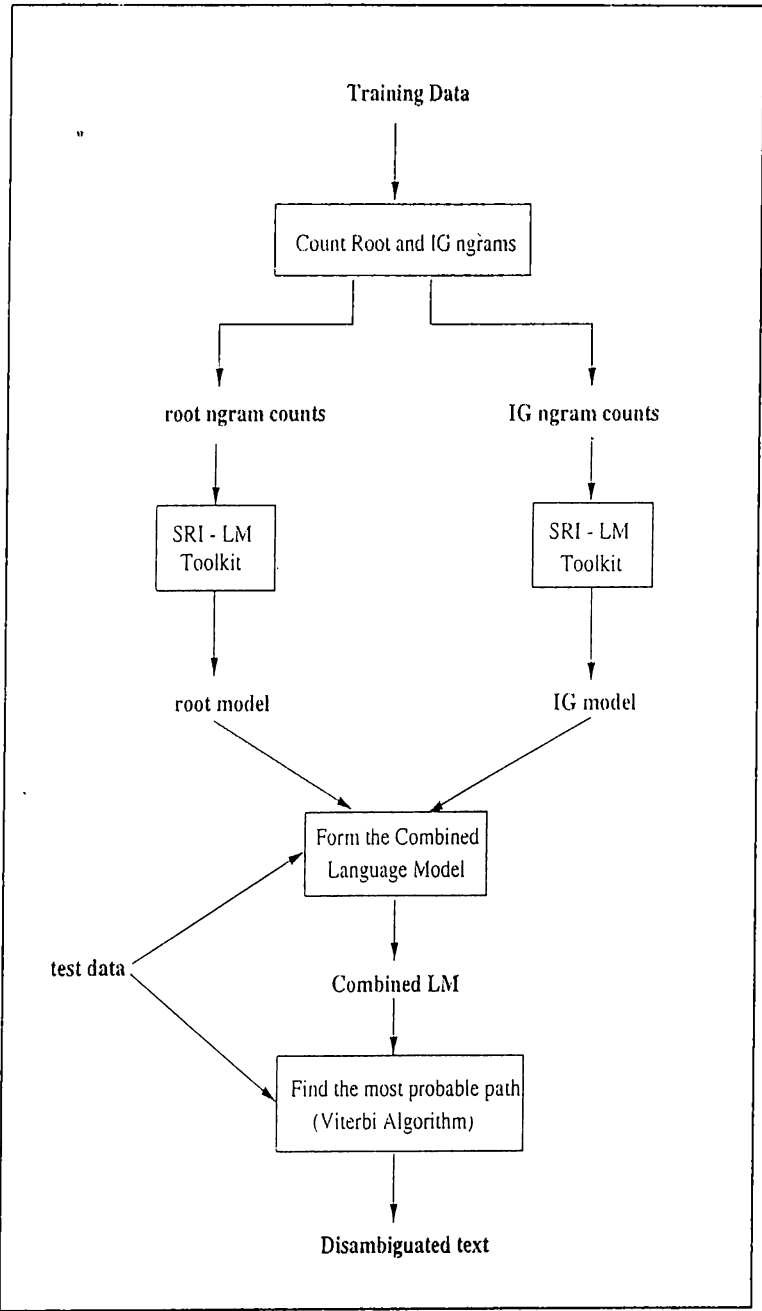


Figure 5.3: Implementation of the  $n$ -gram models.

Sentence:  $\langle S \rangle w_1 w_2 w_3 w_4 \langle /S \rangle$

Parses of  $w_1$ :  $t_{1,1}, t_{1,2}, t_{1,3}$

Parses of  $w_2$ :  $t_{2,1}, t_{2,2}$

Parses of  $w_3$ :  $t_{3,1}$

Parses of  $w_4$ :  $t_{4,1}, t_{4,2}$

HMM:

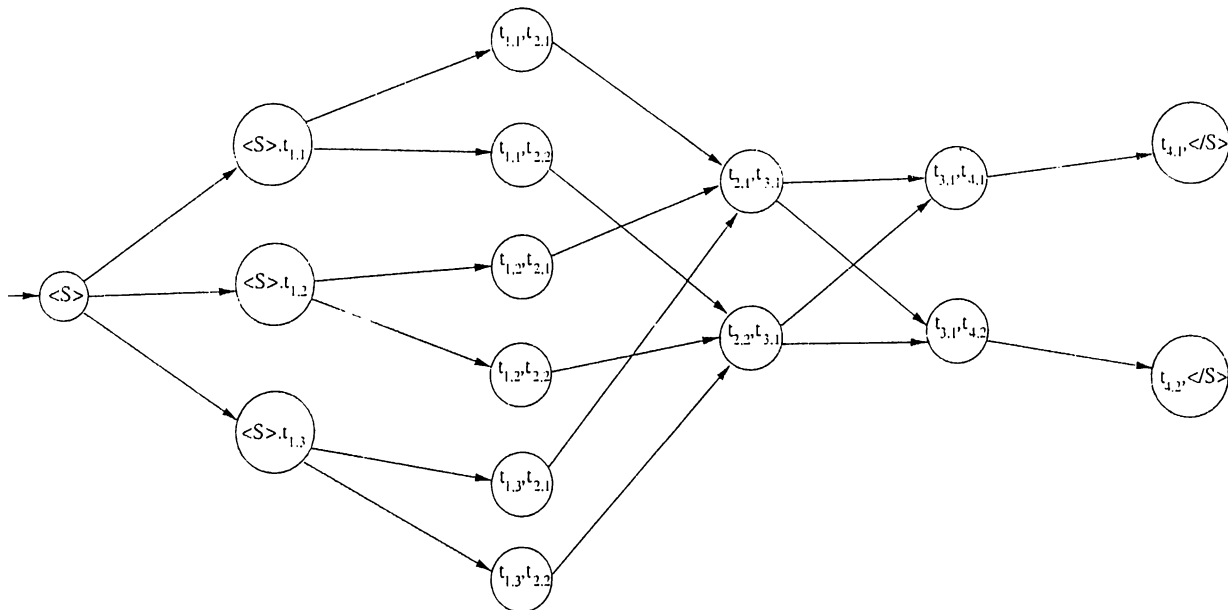


Figure 5.4: The trigram HMM for morphological disambiguation.  $\langle S \rangle$  is the sentence start tag, and  $\langle /S \rangle$  is the sentence end tag.



### 5.3.4 Experiments and Results

To evaluate our models, we first trained our models and then performed morphological disambiguation on our test data.

#### Training and Test Data

Both the test data and training data were collected from the web resources of a Turkish daily newspaper. The tokens were analyzed using the morphological analyzer/generator, developed by Oflazer [1994]. We preprocessed the training and test data, to reduce the morphological ambiguity. The steps of preprocessing are explained in the next section.

The training data consists of the unambiguous sequences (US) consisting of about 650K tokens in a corpus of 1 million tokens, and two manually disambiguated texts of 12,000 and 20,000 tokens. The idea of using unambiguous sequences is similar to Brill's work on unsupervised learning of disambiguation rules for POS tagging [1995b].

#### Preprocessing for Ambiguity Reduction

We preprocess the training and test data to reduce the morphological ambiguity, without reducing accuracy. The following are the steps of preprocessing:

1. We eliminate very rare root words that are ambiguous with a very frequent root word. An example is the word 'bunlar' ('these' in English), which has the following two morphological parses:

(a) bun+Noun+A3pl+Pnon+Nom

(b) bu+Pron+DemosP+A3pl+Pnon+Nom

'bun' is an extremely rare word in Turkish, whereas 'bu' is very frequent, so any parse with the root 'bun' is eliminated.

2. We disambiguate the lexicalized and non-lexicalized collocations involving compound verbs. An example is the compound verb ‘yemek ye-’. For example in the sentence ‘Yemek yenecek’ (in English ‘The dinner will be eaten’), the first word has the following two parses:

- (a)  $yemek+Noun+A3sg+Pnon+Nom$
- (b)  $ye+Verb+Pos^DB+Noun+Inf+A3sg+Pnon+Nom$

and the second word has the following four parses:

- (a)  $ye+Verb^DB+Verb+Pass+Pos+Fut+A3sg$
- (b)  $ye+Verb^DB+Verb+Pass+Pos^DB+Adj+FutPart+Pnon$
- (c)  $yen+Verb+Pos+Fut+A3sg$
- (d)  $yen+Verb+Pos^DB+Adj+FutPart+Pnon$

But, we know that when these words are seen consecutively, the correct parse for the first word is the first parse above, and the correct parse for the second word is the one that is derived from a Verb with root ‘ye’, that is, the one that starts with ‘ye+Verb’.

3. We disambiguate postpositional phrases: Postpositions impose a constraint on the case of the preceding word, some subcategorize for ‘Dative’ noun objects, while others subcategorize for an ‘Ablative’, ‘Nominative’ etc. noun just preceding them. The subcategorization information can be inferred from the type of the postposition. For example, the word ‘sonra’ has the following two parses:

- (a)  $sonra+Postp+PCabl+Temp$
- (b)  $sonra+Adv$

If the preceding word is a noun in Ablative case, then the correct parse is the first one above.

The ambiguity of the training data was reduced from 1.75 to 1.55 using this preprocessor.

The preprocessor analyzes unknown words with an unknown word processor. The unknown words are almost always foreign proper names, words adapted into the language and not in the lexicon, or very obscure technical words. These are sometimes inflected using Turkish word formation paradigms. The unknown word processor assumes that all unknown words have nominal roots. It is constructed in the same way as the morphological analyzer, only the nominal root lexicon recognizes  $S_+$ , where  $S$  is the Turkish surface alphabet.

The test data consists of 2763 tokens, 935 ( $\approx 34\%$ ) of which have more than one morphological analysis after preprocessing. The ambiguity of the test data was reduced from 1.74 to 1.53 after preprocessing.

## Evaluation

As our evaluation metric, we used accuracy, which is the percentage of the correct parses among all selected parses:

$$accuracy = \frac{\# \text{ of correct parses}}{\# \text{ of selected parses}} \times 100$$

The number of selected parses is the number of tokens in our case, since our algorithm selects one parse among the set of possible parses for each token.

There are also other evaluation metrics like recall and precision, but since our system returns only one parse for each token, accuracy results are sufficient.

## Results

The accuracy results are given in Table 5.1. For all cases, our models performed better than baseline tag model. As expected, the tag model suffered considerably from data sparseness. Using all of our training data, we achieved an accuracy of 93.95%, which is 2.57% points better than the tag model trained using the same amount of data. Models 2 and 3 gave similar results. For Model 2, we needed IG 4-gram probabilities, however Model 3 needed only IG 3-gram and

Training Data	Tag Model (Baseline)	Model 1	Model 1 (Bigram)	Model 2	Model 3
US	86.75%	88.21%	89.06%	87.01%	87.19%
US+12,000 words	91.34%	93.52%	93.34%	92.43%	92.72%
US+32,000 words	91.34%	93.95%	93.56%	92.87%	92.94%

Table 5.1: Accuracy results for different models. In the first column, US is an abbreviation for unambiguous sequences.

2-gram probabilities. As can be seen from the results, Model 2 suffered from data sparseness slightly more than Model 3, as expected.

Surprisingly, the bigram version of Model 1 (i.e., Equation (7), but with bigrams in root and IG models), also performs quite well. If we consider just the syntactically relevant morphological features and ignore any semantic features that we mark in morphology, the accuracy increases a bit further. These stem from two properties of Turkish: Most Turkish root words also have a proper noun reading.<sup>4</sup> We count it as an error if the tagger does not get the correct proper noun marking. But this is usually impossible especially at the beginning of sentences where the tagger can not exploit capitalization and has to back-off to a lower-order model. In almost all of such cases, all syntactically relevant morphosyntactic features except the proper noun marking are actually correct. Another important case is the pronoun *o*, which has both personal pronoun (s/he) and demonstrative pronoun readings (it) (in addition to a syntactically distinct determiner reading (that)). Resolution of this is always by semantic considerations. When we count as correct any errors involving such semantic marker cases, we get an accuracy of 95.07% with the best case (cf. 93.91% of the Model 1). This is slightly better than the precision figures that is reported earlier on morphological disambiguation of Turkish using constraint-based techniques [Oflazer and Tür, 1997]. Our results are slightly better than the results on Czech of Hajič and Hladká [1998]. Megyesi [1999] reports a 95.53% accuracy on Hungarian (a language whose features relevant to this task are very close to those of Turkish), with just the POS tags being correct. In our model this corresponds to the root and the POS tag of

<sup>4</sup>In fact, any word form is a potential first name or a last name.

the last IG being correct and the accuracy of our best model with this assumption is 96.07%. When POS tags and subtags are considered, the reported accuracy for Hungarian is 91.94% while the corresponding accuracy in our case is 95.07%. We can also note that the results presented by Ezeiza et al. [1998] for Basque are better than ours. The main reason for this is that they employ a much more sophisticated (compared to our preprocessor) constraint-grammar based system which improves precision without reducing recall. Statistical techniques applied after this disambiguation yield a better accuracy compared to starting from a more ambiguous initial state.

Since our models assumed that we have independent models for disambiguating the root words, and the IGs, we ran experiments to see the contribution of the individual models. Table 5.2 summarizes the accuracy results of the individual models for the best case (Model 1 in Table 5.1.)

Model	Accuracy
IG Model	92.08%
Root Model	80.36%
Combined Model	93.95%

Table 5.2: The contribution of the individual models for the best case.

There are quite a number of classes of words which are always ambiguous and the preprocessing that we have employed in creating the unambiguous sequences can never resolve these cases. Thus statistical models using only these unambiguous sequences as the training data do not handle these ambiguous cases at all. This is why the accuracy results with only unambiguous sequences are significantly lower (row 1 in Table 5.1). The manually disambiguated training sets have such ambiguities resolved, so those models perform much better.

An analysis of the errors indicates the following: In 15% of the errors, the last IG of the word is incorrect but the root and the rest of the IGs, if any, are correct. In 3% of the errors, the last IG of the word is correct but the either the root or some of the previous IGs are incorrect. In 82% of the errors, neither the last IG nor any of the previous IGs are correct. Along a different dimension, in about

51% of the errors, the root and its part-of-speech are not determined correctly, while in 84% of the errors, the root and the first IG combination is not correctly determined.

## 5.4 Morphological Disambiguation of Turkish with Maximum Entropy Models

Maximum Entropy (ME) Modeling is an approach for combining multiple information sources for classification. Recently, ME modeling approach was successfully applied to natural language processing problems, including part-of-speech (POS) tagging [Ratnaparkhi, 1996; Hajič and Hladká, 1998]. Therefore, we decided using ME modeling approach for morphologically disambiguating Turkish text.

In Section 5.2, we show that the problem of morphological disambiguation of Turkish can be reduced as follows, since the morphological parses include the root of the words:

$$\begin{aligned} \operatorname{argmax}_T P(T|W) &= \operatorname{argmax}_T P(T) \\ &= \operatorname{argmax}_T P(t_n|t_1^{n-1}) \times P(t_{n-1}|t_1^{n-2}) \times \dots \times P(t_2|t_1) \times P(t_1) \end{aligned}$$

In the previous section, we approximate the probability of a morphological parse given the previous morphological parses using 3-gram language models. In this section, we approximate this probability as follows:

$$P(t_i|t_1^{i-1}) \propto \prod_{k=1}^{n_i} P(IG_{i,k}|IG_{i-1,n_{i-1}})$$

and use ME models to compute these probabilities. Note that, we are not using the roots in the approximation above. Using the preceding two words as the history would increase (at least double) the number of features. So for simplicity, we condition the probability only on the final IG of the previous word.

### 5.4.1 The Probability Model

The number of possible IGs is very large for classification with ME models. Therefore, we try to reduce the number of classes for the ME models, in a similar way with a recent study for morphologically tagging a similar language, Czech (which is also a free word order, and a highly inflective language.) This study also suggests using the morphological features of the words, as well as the original word forms and part-of-speech tags [Hajič and Hladká, 1998]. The words are analyzed morphologically, and each analysis is seen as a set of 13 morphological features, which can capture the part-of-speech, gender, number, tense etc., information of all the words. Because of the derivational morphology, the analysis of a Turkish word might contain more than one part-of-speech category, so such an approach is not directly applicable. Instead of the whole morphological parses, we represent an IG with 9 categories, namely:

1. Major Part-of-Speech (POS),
2. Minor Part-of-Speech (SUBPOS),
3. Agreement (AGR),
4. Possessive (POSS),
5. Case (CASE),
6. Polarity (POL),
7. Tense-Aspect-Mood Marker 1 (TAM1),
8. Tense-Aspect-Mood Marker 2 (TAM2), and
9. Copula (COP).

The values that these categories can take are listed in Appendix A. Table 5.3 gives examples of how some IGs are represented according to this representation.

IG	1	2	3	4	5	6	7	8	9
Noun+A3sg+P3sg+Nom	Noun	-	A3sg	P3sg	Nom	-	-	-	-
Noun+Inf+A2sg+Pnon+Acc	Noun	Inf	A2sg	Pnon	Acc	-	-	-	-
Verb+Pos+Prog1+A2sg	Verb	-	A2sg	-	-	Pos	Prog1	-	-
Verb+Pos+Fut+Cop+A3sg	Verb	-	A3sg	-	-	Pos	Fut	-	Cop
Verb+Pos+Prog1+Cond+A3sg	Verb	-	A3sg	-	-	Pos	Prog1	Cond	-
Verb+Pass+Pos+Prog1+A3pl	Verb	Pass	A3pl	-	-	Pos	Prog1	-	-
Num+Card	Num	Card	-	-	-	-	-	-	-
Adv+As	Adv	As	-	-	-	-	-	-	-
Pron+PersP+A3sg+Pnon+Abl	Pron	PersP	A3sg	Pnon	Abl	-	-	-	-
Ques+Pres+A1pl	Ques	-	A1pl	-	-	-	Pres	-	-
Adj+AsIf	Adj	AsIf	-	-	-	-	-	-	-

Table 5.3: Examples for representations of IGs using 9 categories. The category that is missing in the IG takes the value “-”.

The probability of an IG, given the final IG of the previous word is computed as the product of the probabilities of the individual category values given the final IG of the previous morphological parse:

$$P(IG_{i,k}|IG_{i-1,n_{i-1}}) = \prod_{j=1}^9 P_j(cat_j(IG_{i,k})|IG_{i-1,n_{i-1}}) \quad (5.20)$$

where  $P_j(x|y)$  is the probability according to the  $j^{th}$  model, and  $cat_j(IG_{i,k})$  is a function which returns the value of the  $j^{th}$  category for  $IG_{i,k}$ , as in the following examples:

$$cat_1(\text{Adj} + \text{AsIf}) = \text{Adj}$$

$$cat_2(\text{Adj} + \text{AsIf}) = \text{AsIf}$$

$$cat_3(\text{Adj} + \text{AsIf}) = -$$

In order to find the probabilities of the individual category values given the final IG of the previous analysis, we construct 9 different models, for each category.

$$P_j(cat_j(IG_{i,k})|IG_{i-1,n_{i-1}}) = \frac{1}{Z_j(IG_{i-1,n_{i-1}})} \prod_{l_j=1}^{m_j} \alpha_{l_j}^{f_{l_j}(cat_j(IG_{i,k}), IG_{i-1,n_{i-1}})} \quad (5.21)$$

where,  $f_{l_j}(cat_j(IG_{i,k}), IG_{i-1,n_{i-1}})$  are the features of our model,  $\alpha_{l_j}$  are the weights of the features, and  $Z_j(IG_{i-1,n_{i-1}})$  is the normalization constant for the history  $IG_{i-1,n_{i-1}}$ .



So, for example:

$$\begin{aligned}
 P(\text{Noun} + \text{A3sg} + \text{P3sg} + \text{Nom} | IG_x) &= P_1(\text{Noun} | IG_x) \times \\
 &P_2(- | IG_x) \times \\
 &P_3(\text{A3sg} | IG_x) \times \\
 &P_4(\text{P3sg} | IG_x) \times \\
 &P_5(\text{Nom} | IG_x) \times \\
 &P_6(- | IG_x) \times \\
 &P_7(- | IG_x) \times \\
 &P_8(- | IG_x) \times \\
 &P_9(- | IG_x)
 \end{aligned}$$

### 5.4.2 Features for Morphological Disambiguation

The conditional probability of a history  $IG_{i-1, n_{i-1}}$  and  $IG_{i,k}$  is determined by the weights whose corresponding features are active (that is, that have the value 1). We tried two types of features for morphologically disambiguating Turkish text. All of these features check the occurrence of category values or IGs with other category values. We have not included features that check the identity of the root or the surface form of the words, in order to keep a small number of features. Our features are described in the following sections.

#### Type-1 Features

The first type of features check the co-occurrence of a category in the previous word's final IG, with another category of the current IG. These features are used in Equation 5.20, and have the following format:

$$f_{l_j}(\text{cat}_j(IG_{i,k}), IG_{i-1, n_{i-1}}) = \begin{cases} 1 & \text{if } \text{cat}_j(IG_{i,k}) = a \text{ and } b \in IG_{i-1, n_{i-1}} \\ 0 & \text{otherwise} \end{cases}$$

Here,  $b \in IG_{i-1, n_{i-1}}$  means that  $IG_{i-1, n_{i-1}}$  has the category value  $b$  in it. An example feature is:

$$f_{1_4}(cat_4(IG_{i,k}), IG_{i-1, n_{i-1}}) = \begin{cases} 1 & \text{if } cat_4(IG_{i,k}) = \text{P3sg} \text{ and } \text{A3sg} \in IG_{i-1, n_{i-1}} \\ 0 & \text{otherwise} \end{cases}$$

and, therefore,

$$f_{1_4}(\text{P3sg}, \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}) = 1$$

$$f_{1_4}(\text{P3sg}, \text{Adj} + \text{AsIf}) = 0$$

$$f_{1_4}(\text{Acc}, \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}) = 0.$$

### Type-2 Features

The second type of features check the co-occurrence of the previous word's final IG with a category of the current IG. These features are used in Equation 5.20, and have the following format:

$$f_{1_j}(cat_j(IG_{i,k}), IG_{i-1, n_{i-1}}) = \begin{cases} 1 & \text{if } cat_j(IG_{i,k}) = a \text{ and } IG_{i-1, n_{i-1}} = b \\ 0 & \text{otherwise} \end{cases}$$

An example feature is:

$$f_{1_4}(cat_4(IG_{i,k}), IG_{i-1, n_{i-1}}) = \begin{cases} 1 & \text{if } cat_4(IG_{i,k}) = \text{P3sg} \text{ and} \\ & IG_{i-1, n_{i-1}} = \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} \\ 0 & \text{otherwise} \end{cases}$$

and, therefore,

$$f_{1_4}(\text{P3sg}, \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}) = 1$$

$$f_{1_4}(\text{P3sg}, \text{Noun} + \text{A3sg} + \text{P3sg} + \text{Nom}) = 0$$

$$f_{1_4}(\text{Acc}, \text{Noun} + \text{A3sg} + \text{Pnon} + \text{Nom}) = 0.$$

### 5.4.3 Training the Models

There are two issues for training the models to use for morphological disambiguation:

1. **Determining the features to use:** We extract the features to use automatically, examining the training data. We use the IG bigrams that occur more than some threshold (that we call as the **IG count threshold**) in the training data for selecting the features.

For the type-1 features, we list the cartesian product of all the feature values (excluding the “-”s) as the candidate features. For type-2 features, we list the pair of IGs (of the first word) and the feature values of the second word (excluding the “-”s again) of the bigrams, as candidate features. From the list of candidate features, we only select the ones that are active in more than a second threshold (that we call as the **feature count threshold**) of the IG bigrams selected from the training data.

2. **Finding the weights of the features:** We use the Generalized Iterative Scaling algorithm [Darroch and Ratcliff, 1972], that is described in Chapter 2, to find the weights of the features. As a stopping criterion for the GIS algorithm (as step 7 of the algorithm given in Section 2.6.6), we use:

$$|E_{\hat{p}}f_i - E_{p^n}f_i| < \epsilon \times E_{\hat{p}}f_i$$

and we call  $\epsilon$  the **threshold weight**.

#### 5.4.4 Testing the Models

Once the features and their weights are computed, we find the probabilities for all possible bigrams in the text to be disambiguated. We then use the Viterbi algorithm [Viterbi, 1967], to find the most probable path, for each sentence, which is returned as the disambiguated sequence.

#### 5.4.5 Experiments and Results

We tested our ME models in a similar framework with the  $n$ -gram models. We performed various experiments to optimize the three threshold values: the IG

count threshold, the feature count threshold and the threshold weight. Our training and test data and these experiments are described in the following subsections.

### Training and Test Data

The training data is the same as the training data we use for training models for morphological disambiguation of Turkish using  $n$ -gram language models. We use two sets of test data for testing our models, the first set (test1) the same as the test data we use for testing our models for morphological disambiguation of Turkish using  $n$ -gram language models and the second set (test2) is a 958 token subset of it. We use the subset for optimizing some of the parameters of our models. The training and test data are preprocessed for ambiguity reduction, as explained in Section 5.3.4.

### Determining the IG Count and Feature Count Thresholds

The number of features that we use for ME modeling is very important, because the running time of the GIS algorithm is dependent on the number of features. Therefore, we tried to limit the number of features, and used thresholds for this purpose. We tried to find optimum values for the thresholds, that result in high accuracy.

Table 5.4 shows the relationship between the thresholds and accuracy using type-1 features. In these 3 experiments, the threshold weight is 0.1. As we increase the number of features, the accuracy increases. Table 5.5 lists the number of features for each model of these 3 experiments.

### Contribution of the Individual Models

In order to evaluate the contribution of the individual category models, we ran an experiment excluding one model (that uses type-1 features) each time. Table 5.6 lists the accuracy values that we obtained with a threshold weight of 0.1.

Test Data	IG Count Threshold	Feature Count Threshold	Accuracy
test2	> 10	$\geq 3$	86.74%
test2	> 10	$\geq 1$	88.93%
test2	> 5	$\geq 1$	89.56%
test1	> 5	$\geq 1$	88.70%

Table 5.4: The accuracy results with models trained varying the counts of the features and IGs to include.

Model	IG Count Threshold>10 Feature Count Threshold $\geq 3$	IG Count Threshold>10 Feature Count Threshold $\geq 1$	IG Count Threshold>5 Feature Count Threshold $\geq 1$
POS	253	376	709
SUBPOS	196	659	851
AGR	114	256	308
POSS	100	226	263
CASE	154	369	431
POL	89	177	193
TAM1	118	276	347
TAM2	72	139	140
COP	59	113	124

Table 5.5: The number of features used for the models, with different threshold value for the features and IGs to include.

Excluding POS, SUBPOS, POL, and COP models did not decrease accuracy. So, we ran another experiment excluding all of these models, and got an accuracy of 89.56%.

Intuitively, POS and SUBPOS models make similar contributions, and using one of them in our model should be sufficient. We ran another experiment excluding only one of them and the other two models POL and COP, and obtained an accuracy of 89.77%, which is the best accuracy, with this threshold. This result supports our intuition, and we excluded only the SUBPOS, POL, and COP models in the remaining experiments.

We trained the 6 models (that we found useful) with the type-1 features,

Model	Accuracy
all - {POS}	89.56%
all - {SUBPOS}	89.77%
all - {AGR}	89.14%
all - {POSS}	85.28%
all - {CASE}	88.30%
all - {POL}	89.35%
all - {TAM1}	88.93%
all - {TAM2}	89.24%
all - {COP}	89.35%
all	89.35%
all - {COP.POL,SUBPOS,POS}	89.56%
all {COP.POL.SUBPOS}	89.77%

Table 5.6: The accuracy results for a threshold weight of 0.1. “all” includes 9 models. The test set used for these experiments is test2.

decreasing the threshold weight to 0.01 This also increased the accuracy. Table 5.7 lists the accuracy obtained with models trained using a threshold weight of 0.01. The best accuracy with this threshold, which is also the best accuracy we obtained using ME models, is 90.60%, and is obtained using the 6 individual category models. The accuracy corresponding to the same experiment with a threshold weight of 0.1 was 89.77%.

Test Data	Model	Accuracy
test2	all - {POS}	83.82%
test2	all - {AGR}	86.01%
test2	all - {POSS}	88.51%
test2	all - {CASE}	87.58%
test2	all - {TAM1}	83.08%
test2	all - {TAM2}	89.45%
test2	all	90.60%
test1	all	89.57%

Table 5.7: The accuracy results for a threshold weight of 0.01. “all” includes the remaining 6 models.

### Type-2 Features

In our experiments with the type-2 features, we only used the (useful) 6 models. The IG count threshold was 5, the feature count threshold was 1. Table 5.8 lists the number of type-2 features for these experiments and Table 5.9 lists the accuracy values. The best accuracy that we obtained using type-2 features is 89.77%, which is the same for threshold weights of 0.1 and 0.01.

Model	Number of Features
POS	1383
AGR	677
POSS	636
CASE	846
TAM1	678
TAM2	430

Table 5.8: The number of type-2 features used for the models, with IG Count  $>5$  and Feature Count Threshold  $\geq 1$ .

Test Data	Threshold Weight	Accuracy
test2	0.1	89.77%
test2	0.01	89.77%
test1	0.1	89.57%
test1	0.01	89.93%

Table 5.9: The accuracy results with Type-2 features. Only the IG bigrams that occurred more than 5 times were used when finding the features.

## 5.5 Discussion

The best accuracy that results with the  $n$ -gram modeling approach is 93.95%, and with the ME modeling approach is 89.93%, with the same training and test data.

In terms of accuracy, ME models perform worse, because of the following reasons:

- **The number of classes is very large.** Even if we used tags instead of categories of features, the number of classes that we are trying to model is very large. This causes a sparsity problem. Furthermore, the running time of the GIS algorithm is (linearly) dependent on the number of classes. Therefore, we make too many assumptions to reduce the number of classes. For example, we represented IGs using 9 categories, and assumed the probability of an IG is the product of the probabilities of the values of these 9 categories.
- **The number of features is very large.** The running time of the GIS algorithm is also dependent on the number of features. Therefore, we tried to reduce the number of possible features. For example, we introduced two thresholds, one for reducing the number of IG bigrams to use for inducing the candidate features, and one for the count of the features to include in the models. We also checked for only very simple features. This may also be a reason for the poor performance. One can include more complex features, which can check for multiple category values.
- **We made too many approximations to simplify the models.** We assume that the IGs depend only on the final IG of the previous word for the ME models, otherwise the number of features to use would be much larger. So, our ME models still have the weakness of the bigram language models. We also assume that the values of categories in an IG is independent of the values of other categories' values.
- **Roots are not included in the models.** With the  $n$ -gram modeling approach, we have found that roots are very useful for morphological disambiguation. We obtained a 1.87% absolute accuracy improvement by combining the IG model probabilities with the root model probabilities. However, we have not included roots when using maximum entropy models, in order to simplify our models, by reducing the number of parameters.



On the other hand, our results are consistent with those of Brants [2000], who found that  $n$ -gram models outperform ME models for part of speech tagging of English.

In terms of training speed,  $n$ -gram models are again better. Training  $n$ -gram models with MLE is much faster than training ME models with the GIS algorithm.

# Chapter 6

## Application to Speech Recognition

### 6.1 Introduction

Speech is the most natural way of communication for human beings, therefore the use of speech as a way of communication between people and computers is very important. Hence, recognition of speech by computers is very important. Speech recognition is the problem of finding the sequence of words  $W = w_1^n = w_1, w_2, \dots, w_n$  corresponding to a sequence of acoustic signals  $A = a_1^m = a_1, a_2, \dots, a_m$ . In the past 20 years, significant advances have been made in speech recognition, and many successful systems have been developed [Woodland *et al.*, 1999; Davenport *et al.*, 1999; Wegmann *et al.*, 1999, among others].

The aim of speech recognition is to find the sequence  $W^*$ , that maximizes  $P(W|A)$ :

$$\begin{aligned} W^* &= \operatorname{argmax}_W P(W|A) \\ &= \operatorname{argmax}_W \frac{P(A|W) \times P(W)}{P(A)} \end{aligned}$$

$$= \operatorname{argmax}_W P(A|W) \times P(W) \quad (6.1)$$

For a given acoustic signal  $A$ ,  $P(A|W)$  is estimated by an acoustic model and  $P(W)$  is estimated by a language model [Jelinek, 1998].

The most popular evaluation metric for speech recognition systems is *word error rate* (WER), which is the percentage of the incorrectly recognized words. So the aim of a speech recognizer is to produce output with a low WER.

An alternative to the WER is *accuracy*, which is the percentage of the correctly recognized words:

$$\text{accuracy} = (100 - \text{WER})\%$$

Decreasing the WER is the same as increasing the accuracy. We use both of these metrics for evaluating our methods.

In this chapter, we explain our work on reorganizing the output of a speech recognizer in order to increase its accuracy. In the next section, we describe the recognizer that we use, then we elaborate on the problem that is the subject of this chapter. We describe how we approximate the acoustic models and present different approaches for language modeling. We conclude with our experiments and results and their analysis.

## 6.2 The Recognizer

The speech recognizer we use is an isolated word, large vocabulary speech recognition system for Turkish, developed as an M.Sc. thesis, at Bilkent University [Yilmaz, 1999]. An isolated word speech recognition system requires the speaker to make a pause of a reasonable amount of duration between consecutive words. The alternative to this is a continuous speech recognition system, for which the speakers utter their sentences in a natural manner, like in the real life.

The recognizer uses three state left-to-right hidden Markov models to model the triphones (sequences of three phones) of the uttered words. The HMMs for

words is constructed by appending the HMMs of the triphones making up the words, as described by Young [1995].

This recognizer uses only the acoustic model probabilities to recognize speech, that is, it tries to estimate the spoken words using only the probabilities  $P(A|W)$ , and does not utilize any language model information. It outputs a 10-best list for each word, in the order of decreasing probability, but the probability values are not listed.

### 6.3 Problem

In this chapter, we describe our work on rescoreing the  $n$ -best list output of a speech recognizer, in order to get better estimates for the word sequence  $W$ . Our aim is to reduce the WER. The  $n$ -best list is the most probable  $n$  word sequences, that the speech recognizer outputs, given the acoustic signal sequence  $A$ .

We form a lattice for each sentence, using these 10-best lists, and try to make better estimates for the uttered sentences, incorporating the language model probabilities. We use the Viterbi algorithm [Viterbi, 1967], to find the most probable path through the lattice, where the probability of a path is computed using both acoustic and language model probabilities. Figure 6.1 gives an example of the lattice construction from a 3-best list, which is the speech recognizer output, and the rescoreing process through this lattice.

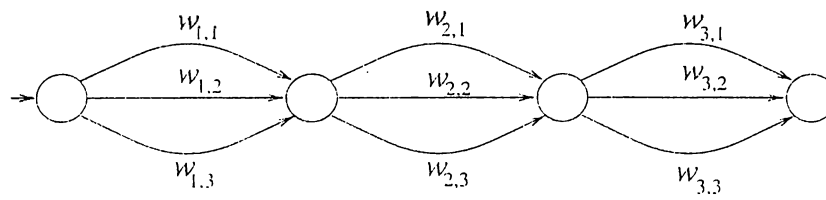
### 6.4 Approximating the Acoustic Model Probabilities

As we mentioned in the previous sections, the recognizer that is available to us lacks the acoustic model probabilities for the  $n$ -best list, but outputs the  $n$  most probable words in a decreasing order. Therefore, we needed to approximate the acoustic model probabilities to use for rescoreing the  $n$ -best lists. We tried three

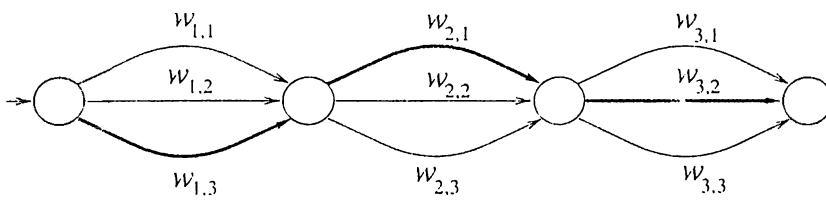
(a) The speech recognizer output (3-best list for each word):

$w_1$	$w_2$	$w_3$
$w_{1,1}$	$w_{2,1}$	$w_{3,1}$
$w_{1,2}$	$w_{2,2}$	$w_{3,2}$
$w_{1,3}$	$w_{2,3}$	$w_{3,3}$

(b) The corresponding lattice:



(c) The best path through the lattice:



(d) The output:

$w_{1,3} \quad w_{2,1} \quad w_{3,2}$

Figure 6.1: The  $n$ -best list rescoring process.

approaches for this purpose, and these are explained in detail, in the following sections.

### 6.4.1 Equal Probabilities

The first approach assumes that, all the words in the 10-best list are equally probable. This is a weak assumption, since we already know that the words are ordered according to decreasing probability. However, this can be seen as a baseline in order to demonstrate the contribution of the language model alone.

### 6.4.2 Linearly Decreasing Probabilities

The second approach assumes that the acoustic model probabilities for the words in the 10-best list should decrease linearly, that is, if the first word is assigned an acoustic model probability of  $10x$ , then the second word is assigned an acoustic model probability of  $9x$ , the third word is assigned an acoustic model probability of  $8x$ , and the last word is assigned an acoustic model probability of  $x$ , as shown in Figure 6.2.

### 6.4.3 Exponentially Decreasing Probabilities

The third approach assumes that the acoustic model probabilities for the words in the 10-best list should decrease exponentially, that is, if the first word is assigned an acoustic model probability of  $10x$ , then the second word is assigned an acoustic model probability of  $\frac{10}{2}x$ , the third word is assigned an acoustic model probability of  $\frac{10}{2^2}x$ , and the last word is assigned an acoustic model probability of  $\frac{10}{2^9}x$ , as shown in Figure 6.3.

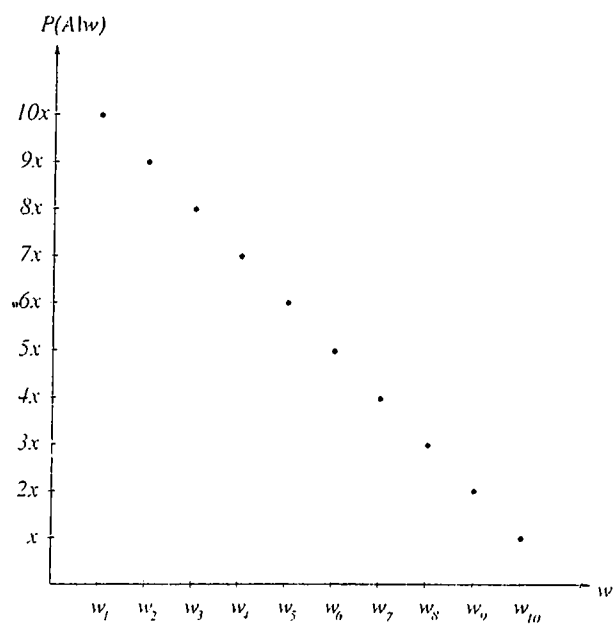


Figure 6.2: Approximating acoustic model probabilities with a linear function.

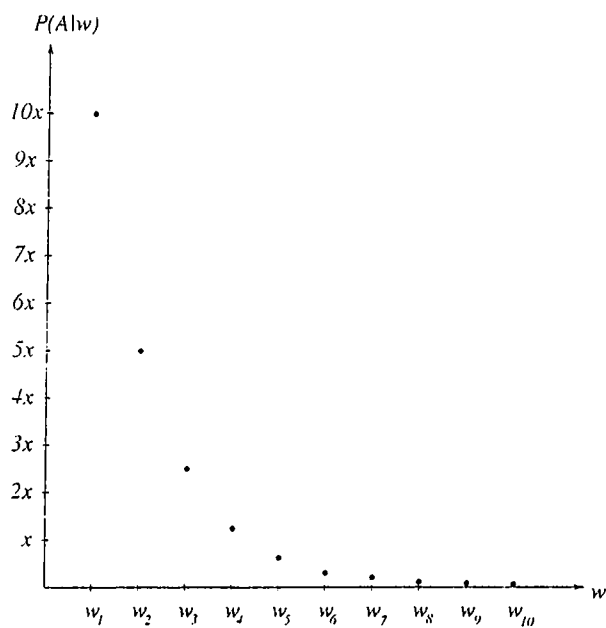


Figure 6.3: Approximating acoustic model probabilities with an exponential function.

## 6.5 Language Modeling

In order to estimate the language model probabilities for a sequence of Turkish words, we constructed three different language models, a word-based language model (similar to the language modeling approaches for English), an IG-based language model, and a prefix-suffix language model. The IG-based model is a consequence of our models for morphological disambiguation and the prefix-suffix language modeling is an approximation of the IG-based modeling approach. These models are described in detail in the following sections.

### 6.5.1 Word-Based Language Modeling

For word-based language modeling, we constructed trigram language models where the probability of a word given the previous words is approximated by the probability of this word given only the previous two words. The language model probabilities are computed as follows:

$$\begin{aligned} P(W) &= \prod_{i=1}^n P(w_i | w_1^{i-1}) \\ &\propto \prod_{i=1}^n P(w_i | w_{i-2}, w_{i-1}) \end{aligned} \quad (6.2)$$

This approach has been successfully used for languages like English in various natural language and speech processing tasks.

The trigram language models are constructed using the SRILM Toolkit, which smooths the probabilities using Good-Turing modeling [Gale, 1994] combined with Backoff smoothing technique [Katz, 1987].



### 6.5.2 IG-Based Language Modeling

Part-of-speech tags or morphological parses can be used as the basis of equivalence classes for language modeling [Heeman and Allen, 1997; Niesler and Woodland, 1996; Bangalore, 1996], as in class-based language modeling. The typical approach for using part-of-speech tags in language modeling is to sum the probabilities over all of the part-of-speech tag possibilities. So, the probability of a word sequence is computed as follows, using the trigram sequences:

$$\begin{aligned}
 P(W) &= \sum_{t_1^n} P(w_1^n, t_1^n) \\
 &= \sum_{t_1^n} \prod_{k=1}^n P(w_k | w_1^{k-1}, t_1^k) \times P(t_k | w_1^{k-1}, t_1^{k-1}) \\
 &\approx \sum_{t_1^n} \prod_{k=1}^n P(w_k | t_k) \times P(t_k | t_{k-2}, t_{k-1}) \quad (6.3)
 \end{aligned}$$

where  $t_1^k$  is the sequence of part-of-speech tags, or morphological parses in our case.

Heeman [1997] suggests another approach, redefining the speech recognition problem, so that it finds the best word and part-of-speech tag sequence. Let  $T$  be the part-of-speech tag sequence for the word sequence  $W$ . The goal of speech recognition is now to solve the following:

$$\begin{aligned}
 W^* T^* &= \operatorname{argmax}_{W, T} P(W, T | A) \\
 &= \operatorname{argmax}_{W, T} \frac{P(A | W, T) \times P(W, T)}{P(A)} \\
 &= \operatorname{argmax}_{W, T} P(A | W, T) \times P(W, T) \quad (6.4)
 \end{aligned}$$

The first probability  $P(A | W, T)$  is again computed by the acoustic model, which traditionally excludes the part-of-speech category assignment. In fact, this probability can be reasonably approximated by  $P(A | W)$ . The second probability  $P(W, T)$  is computed by the part-of-speech based language model, and this accounts for both the sequence of words and the part-of-speech tag assignment for those words. The  $P(W, T)$  can be rewritten as follows:

$$P(W, T) = \prod_{k=1}^n P(w_k, t_k | w_1^{k-1}, t_1^{k-1})$$

$$\begin{aligned}
&= \prod_{k=1}^n P(w_k | w_1^{k-1}, t_1^k) \times P(t_k | w_1^{k-1}, t_1^{k-1}) \\
&= \prod_{k=1}^n P(t_k | w_1^{k-1}, t_1^{k-1})
\end{aligned} \tag{6.5}$$

$P(w_k | w_1^{k-1}, t_1^k) = 1$  since we use the full morphological parses as part-of-speech tags. We then approximate  $P(t_k | w_1^{k-1}, t_1^{k-1})$  by  $P(t_k | t_1^{k-1})$  and then compute  $P(W, T)$ , using trigram models as follows:

$$P(W, T) = \prod_{k=1}^n P(t_k | t_{k-2}, t_{k-1}) \tag{6.6}$$

We then use the root and IG probabilities to compute the probability of the parse sequence as in the morphological disambiguation approach.

For the IG-based language modeling approach, we first analyze all the words in the  $n$ -best list morphologically, using the morphological analyzer/generator, developed by Oflazer [1994]. This process increases the number of candidates in the 10-best list, since some of these words are morphologically ambiguous. We assign the same acoustic model probability to all of the morphological parses of a word and this probability is the acoustic model probability of the surface form of the word. We then form the lattice for each sentence, using the extended set of candidates and rescore the lattice using the acoustic model and IG-based language model probabilities. Once the most probable path through the lattice is selected, the surface forms of all of the parses on this path are generated using the same toolkit.

Using the IG-based language modeling approach, we can also morphologically disambiguate the recognized sequence of words, using our statistical models. However, the morphological analysis and generation processes are the extra steps required in order to use IG-based language models.

### 6.5.3 Prefix-Suffix Language Modeling

Prefix-Suffix language modeling is an approximation of the IG-based language modeling approach, in order to get rid off the morphological analysis and generation steps. As I already mentioned, Turkish is an extremely suffixing language, and looking at the suffixes of the words, we can clearly see some patterns. For example, in a noun-noun NP, with an owner relationship, the first word gets a genitive case marker, and the second word is marked with a possessive suffix that agrees with the first word:

*çocuğun*      *kalemi* (the pencil of the child)  
 child+GEN    pencil+P3SG

*okulun*      *yeri* (the location of the school)  
 school+GEN    location+P3SG

Though, the surface forms of these suffixes may change as a result of vowel harmony:

*evin*      *terası* (the terrace of the house)  
 house+GEN    terrace+P3SG

these patterns may still be captured by a statistical model.

In this model, the initial part of a word corresponds to the root, and the final part corresponds to the suffix appended to the root. Figure 6.4 shows the dependence of the letter sequences.

This model approximates the probability of a word given the previous two words by the product of the probability of the initial part of the word given the initial parts of the previous two words and the probability of the final part of the word given the final parts of the previous two words:

$$\begin{aligned} P(W) &= \prod_{i=1}^n P(w_i | w_1^{i-1}) \\ &= \prod_{i=1}^n P(w_i | w_{i-2}, w_{i-1}) \end{aligned}$$

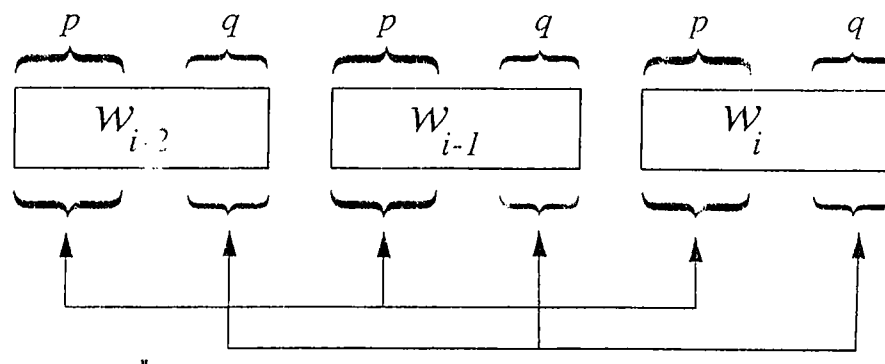


Figure 6.4: Dependence of the letter sequences.

$$= \prod_{i=1}^n P(\text{Pre}_p(w_i) | \text{Pre}_p(w_{i-2}), \text{Pre}_p(w_{i-1})) \times P(\text{Suf}_q(w_i) | \text{Suf}_q(w_{i-2}), \text{Suf}_q(w_{i-1})) \quad (6.7)$$

$\text{Pre}_p(w_i)$  is a function that returns the initial  $p$  letters of the word  $w_i$ , and  $\text{Suf}_q(w_i)$  is another function that returns the final  $q$  letters of the word  $w_i$ . If the word  $w_i$  is shorter than  $p$  characters, then  $\text{Pre}_p(w_i)$  returns the whole word. Likewise, if it is shorter than  $q$  characters, then  $\text{Suf}_q(w_i)$  returns the whole word. The parameter  $p$  corresponds to the length of the root part and  $q$  corresponds to the length of the suffix part of the words.

We tried various experiments changing the values of the parameters  $p$  and  $q$ . Examining 1 million words of morphologically disambiguated Turkish text, we found out that the average root length is 4.10 letters, and the average suffix length is 1.71 letters.

## 6.6 Experiments and Results

In order to test our language models, we trained the speech recognizer, and formed an 10-best list for each word in our test data, and then constructed a lattice for each sentence. We returned the sequence of words on the most probable path, according to each model, as the recognized sentence.

In order to modify the contribution of the acoustic and language models, we

introduced two weights,  $\alpha$  and  $\beta$ . Then, the formulation of the problem changes as follows:

$$\begin{aligned} W_* &= \operatorname{argmax}_W P(W|A) \\ &= \operatorname{argmax}_W P(A|W) \times P(W) \end{aligned} \quad (6.8)$$

$$= \operatorname{argmax}_W P(A|W)^\alpha \times P(W)^\beta \quad (6.9)$$

$\alpha$  is called as the **acoustic model weight**, and  $\beta$  is called as the **language model weight**. For the word based language modeling approach, there is no constraint on the weights  $\alpha$  and  $\beta$ . However, for the IG-based and prefix-suffix language modeling approaches,  $\alpha$  and  $\beta$  satisfies the following constraint:

$$\alpha + \beta = 1 \quad (6.10)$$

This is because, for the word-based language modeling approach, we only used the SRILM toolkit. However, for the other two approaches, we used SRILM toolkit, only when constructing the root and IG models (for IG-based modeling) and the word-initial and word-final sequence models (for prefix-suffix modeling). For combining the individual models, we developed another program. Note that the two weighting schemes are the same, since one can always scale the weights of the models, so that their sums will be 1.

In all our experiments, we tried using equal weights for the acoustic and language models. These weights are 1 when word-based language models are used, and 0.5 when IG-based or prefix-suffix language models are used. We also tried to find the optimum weights.

In the following sections, we describe the training and test data and our experiments.

### 6.6.1 Training and Test Data

Our training data for the speech recognizer consists of 535 words which cover all the phone trigrams in our test data. Each word in the training set is uttered three times, by a female speaker.

Accuracy	60.83%
Best Possible Accuracy	79.41%

Table 6.1: The initial performance of the recognizer.

The training data for the word-based and prefix-suffix language models are 1 million and 20 million words, collected from the web pages of a Turkish daily newspaper. Around 60% of the numbers in these corpora are converted into their vocalized versions.

The training data for the IG-based language model is the one that we used for forming IG-based language models we used for training the morphological disambiguation system.

The test data consists of 1361 words, and the vocabulary size of our test data is 773 words.

### 6.6.2 Initial Performance of the Recognizer

Table 6.1 lists the initial accuracy results of the recognizer with our test data. When only the acoustic model is used, the accuracy is 60.83%. This number is obtained assuming that the first candidate in the 10-best list is returned as the recognized word. 79.41% of the words have the correct form in the 10-best list. Therefore, this is the best possible accuracy that can be obtained when we rescore the 10-best list by incorporating the language model.

### 6.6.3 Word-Based Language Modeling

For word-based language modeling, we used traditional 3-gram language models. Table 6.2 lists the accuracy when we use the word-based language models trained with 1 million and 20 million words. The third column of the table lists the accuracy obtained with no weighting (that is, the  $\alpha$  and  $\beta$  weights have value 1).

LM Training Data Size	Acoustic Model	Accuracy (No Weighting)	Accuracy (With Weights)
1 Million Words	Equal	43.43%	-
	Linear	66.68%	68.15% ( $\alpha = 1.2, \beta = 1$ )
	Exponential	59.49%	67.42% ( $\alpha = 2.3, \beta = 1$ )
20 Million Words	Equal	51.03%	-
	Linear	59.36%	61.09% ( $\alpha = 1.2, \beta = 1$ )
	Exponential	55.42%	61.09% ( $\alpha = 2.3, \beta = 1$ )

Table 6.2: The performance of the recognizer after rescoreing the  $n$ -best list with a word based LM.

The fourth column lists the accuracy obtained with optimized weights.<sup>1</sup> Since the toolkit works very slow with training data of 20 million words, we have not optimized the weights for this training data, instead used the optimum weights obtained using training data of 1 million words.

The best accuracy that we achieved using word-based language models with equal weights is 66.68%, and with optimized weights is 68.15%. Note that the accuracy of the speech recognizer is reduced when we approximate the acoustic model probabilities with equal probabilities.

#### 6.6.4 IG-Based Language Modeling

IG-based language models gave the worst results for the  $n$ -best list rescoreing task. The accuracy numbers obtained with these models is listed in Table 6.3. All approximations to acoustic model probabilities with equal weights for acoustic and language models reduced the accuracy of our speech recognizer. However, small gains over the baseline accuracy (that is, the initial accuracy of the recognizer) could be acquired optimizing the acoustic and language model weights.

The run-time of the system using IG-based language models is longer than

---

<sup>1</sup>The weights are given in parentheses.

Acoustic Model	Accuracy (No Weighting)	Accuracy (AM Weight)
Equal	29.31%	-
Linear	41.37%	63.22% (0.03)
Exponential	58.42%	64.35% (0.14)

Table 6.3: The performance of the recognizer after rescoreing the  $n$ -best list with an IG based LM. The AM Weight stands for the Acoustic Model Weight.

LM Training Data Size	Acoustic Model	Accuracy (No Weighting)	Accuracy (AM Weight)
1 Million Words	Equal	35.24%	-
	Linear	67.29%	67.88% (0.56)
	Exponential	66.88%	67.62% (0.43)

Table 6.4: The performance of the recognizer after rescoreing the  $n$ -best list with a prefix-suffix LM, which has a root size of 3 letters and suffix size of 2 letters.

the run-time of the systems using the other models, because of the extra morphological analysis and generation steps. The increase in the number of candidates in the  $n$ -best list, introduced as a result of morphological ambiguity is another factor, increasing the run-time of this system.

### 6.6.5 Prefix-Suffix Language Modeling

We acquired the best results with the prefix-suffix language models. We tried 3 versions of these models, modifying the length of the root and the suffix parts:

1. **3-letter prefix, 2-letter suffix approximation:** The accuracy numbers that we acquired with 3-letter root and 2-letter suffix approximation is listed in Table 6.4. The best accuracy achieved using this model with equal acoustic and language model weights is 67.29%, and with optimized weights is 67.88%. Both are obtained when we used linearly decreasing probabilities for approximating the acoustic models.



LM Training Data Size	Acoustic Model	Accuracy (No Weighting)	Accuracy (AM Weight)
1 Million Words	Equal	40.37%	-
	Linear	67.75%	68.89% (0.61)
	Exponential	67.75%	68.82% (0.39)

Table 6.5: The performance of the recognizer after rescoring the  $n$ -best list with a prefix-suffix LM, which has a root size of 4 letters and suffix size of 2 letters.

LM Training Data Size	Acoustic Model	Accuracy (No Weighting)	Accuracy (AM Weight)
1 Million Words	Equal	42.10%	-
	Linear	69.08%	69.82% (0.57)
	Exponential	68.28%	69.95% (0.32)
20 Million Words	Equal	46.43%	-
	Linear	70.61%	71.35% (0.53)
	Exponential	69.42%	70.95% (0.42)

Table 6.6: The performance of the recognizer after rescoring the  $n$ -best list with a prefix-suffix LM, which has a root size of 4 letters and suffix size of 3 letters.

2. **4-letter prefix, 2-letter suffix approximation:** We achieved better accuracy results when we increased the length of the sequence approximating the root. These results are listed in Table 6.5. The accuracy for linearly decreasing and exponentially decreasing acoustic model probabilities, with equal weights for acoustic and language models is the same, and 67.75%. When we optimized the model weights, we obtained an accuracy of 68.89% with linearly decreasing acoustic model probabilities.
3. **4-letter prefix, 3-letter suffix approximation:** The best accuracy results are obtained with a root length of 4 letters and a suffix length of 3 letters, as listed in Table 6.6. Since this model gave the best results with a training data of 1 million words, we also tried using training data of 20 million words, and achieved our best results for  $n$ -best list rescoring. The best accuracy that we achieved by  $n$ -best list rescoring is 71.35%, and is obtained using linearly decreasing acoustic model probabilities, and optimized.

$\lambda$	Accuracy
0.03	45.26%
0.06	52.31%
0.15	64.07%
0.5	64.91%

Table 6.7: The effect of exponentially decreasing acoustic model probabilities on performance.

model weighting.

In our experiments with our best prefix-suffix models, we also tried changing the approximation function for acoustic model probabilities. We used exponentially decreasing probabilities, which decay slower than our initial exponential approximation, so the probability of the  $i^{\text{th}}$  word in the  $n$ -best list is estimated by the following function:

$$\frac{1}{x} \times e^{-\lambda i}$$

where  $x$  is a normalization constant,  $i$  is the order of the word, and  $\lambda$  is a parameter to control the decay. Table 6.7 lists our results with various  $\lambda$  values.

We made our experiments with 4 letter prefix and 3-letter suffix models, which gave the best performance, for our task, which was 68.28% with our exponential function. As can be seen from the new results, the accuracy gets better as we increase  $\lambda$  for exponential functions, which means that we get better results when the probabilities for the words in the  $n$ -best list decrease faster, which is also consistent with our initial exponential approximation. We obtained similar results, when we changed our linear function, too.

## 6.7 Discussion

In rescoreing the  $n$ -best list output of a speech recognizer in order to increase its accuracy, the prefix-suffix language models, which we proposed as an approximation to IG-based models outperformed all our models. The reason that IG-based

models performed poorly is that ambiguity is introduced by morphological analysis. Morphological parses of the words are ambiguous, and since we do not know which one is the correct parse in that context, we are using all of the parses, which complicates the selection procedure.

Prefix-suffix models eliminated the ambiguity problem of IG-based models, since we are using fixed lengths that only approximate the root and the suffix parts.

However, there is still room for further improvement, so clustering and real stem+suffix models, which are described in the concluding chapter, can be investigated as a future work.

# Chapter 7

## Application to Spelling Correction

### 7.1 Introduction

The aim of a spelling checker is to find words in a text which are mis-spelled and return a set of possible candidates for the correct version of those words, and the goal of a spelling corrector is to find and return the correct version of the mis-spelled word, among the possible candidates. most of the time using the context in which the word occurs. Spelling checkers and correctors are a part of all modern word processors and are also important in applications like optical character recognition and hand writing recognition.

In this chapter, we describe the application of our language modeling approaches for context dependent spelling error correction. In a similar way to  $n$ -best list rescoring for speech recognition, we form a lattice using the correct words and the spelling corrector candidates for the mis-spelled words. We then use the language model to select the most probable path through the lattice, which we return as the spell corrected version of the text.

We use the spelling checker developed by Oflazer [1996]. The spelling checker

produces as output an unordered list of candidate words, which might be the correct version of the mis-spelled word. All of the candidates have equal probability (which is not true in real applications).

In the next section, we describe the types of spelling errors with examples. Then, we describe how we use the previously described language modeling approaches for this task and we conclude with experimental results and discussion.

## 7.2 Spelling Errors

Spelling errors can be classified into 4 types [Jurafsky and Martin, 1999]:<sup>1</sup>

1. **Insertion:** an extra letter is inserted into the word.
2. **Deletion:** one of the letters of the word is omitted.
3. **Substitution:** one of the letters of the word is mis-spelled, that is another letter is typed instead of the correct letter.
4. **Transposition:** two consecutive letters of a word are interchanged.

Examples to the 4 types of spelling errors for English and Turkish are given in Table 7.1.

Note that spelling correction in a language like Turkish, which has a complex morphology is not always an easy task. Sometimes the error might be on the suffixes of the word, or longer words might have more than one spelling error, as in the following example:

**Mis-spelled word:** `uygarlaştıramadıklarımız`

**Correct Word:** `uygarlaştıramadıklarımız`

---

<sup>1</sup>We do not consider the words that are pronounced very similarly, but spelled in a different way, such as *principle* and *principal* or *whether* and *weather* as in Mangu and Brill [Mangu and Brill, 1997]. We do not treat a word's surface form which corresponds to another word's surface form as mis-spelled.

Error type	Language	Correct Word	Mis-spelled Word
Insertion	English	table	tabale
	Turkish	evlerdeki	evlerideki
Deletion	English	bird	brd
	Turkish	çalışkan	çlışkan
Substitution	English	house	hoube
	Turkish	önündeki	onündeki
Transposition	English	school	scholo
	Turkish	tirbün	tirbün

Table 7.1: Examples of 4 types of spelling errors for English and Turkish.

In the example above, there is one substitution and one transposition mistake. The letters which cause the error are shown in italic in the mis-spelled word.

If the mis-spelled version of a word is also a grammatical word in the language, the spelling checker does not search for its correct versions. For example, the spell-checker does not classify the word ‘tale’ as mis-spelled, even though it is the mis-spelled version of the word ‘table’. Similarly, the word ‘evdekilerin’ might be mis-spelled as ‘evdekileri’, which is also a Turkish word, and the spelling corrector does not classify this word as mis-spelled and so does not try to find the possible correct candidates.

### 7.3 Language Modeling

We use the language model probabilities in order to select the best path through the lattice, which is formed using the words of the text and the candidates of a spelling corrector for the mis-spelled words. For spelling checking, we used the word-based and prefix-suffix language models that we used  $n$ -best list rescoring for speech recognition. We again used the word-based models as a baseline, and we preferred prefix-suffix language models, since they resulted in better results in the previous task. We set the prefix length to 4 letters and the suffix length to 3 letters, since these lengths gave the best results in the previous task.

In the spelling correction, we do not have prior probabilities for the possible candidates, since the spelling checker does not output probabilities for each word or an ordered list. So, we assume that each candidate word is equally likely, and assign each of them an equal probability. We use these probabilities as the acoustic model probabilities of the  $n$ -best list lattice, and compute the most probable word sequence in the same way as described in the previous chapter.

## 7.4 Experiments and Results

We made a variety of experiments for spelling correction. In our experiments, we used accuracy as our evaluation metric, as in the previous tasks.

In the following sections, we describe these experiments and their results, along with the test and training data that we used.

### 7.4.1 Training Data

The training data that we used for spelling correction is the same set as we used for training the word-based and prefix-suffix language models for speech recognition. We used two corpora of 1 million and 20 million words of newspaper text.

### 7.4.2 Test Data

The test data that we used is also the same text of 1361 words that we used for  $n$ -best list rescoring. We first distorted the test data, assuming equal probabilities for the 4 types of spelling errors. The distortion software required only two parameters, the percentage of the mis-spelled words, and what percentage of those words contain two errors.

During distortion, we randomly selected the words that contain the errors, and

Test File	Error Percentage	Double Error Percentage
1	10%	10%
2	10%	20%
3	20%	10%
4	20%	20%

The Double Error Percentage is the percentage of double errors in a word among the mis-spelled words.

Table 7.2: The variations of our test data.

the location of the error inside the word. Also, for insertion and substitution, we randomly selected the new letter, so we did not consider the locations of the letters on the keyboard, or the similarity of the shapes of the letters.

For simplification, we assumed that each word can have at most two errors. Table 7.2 lists our test files and their properties, that is their total error rates and double error rates.

Then we used a spelling checker in order to find possible candidates for mis-spelled words. The spelling checker gets the number of mistakes to correct in each word, as a parameter. Since we distorted the data so that each word can have at most two errors, we also set the spell checker parameter to two.

### 7.4.3 Results

The results of spelling correction are listed in Table 7.3. For each file, we used two language models, each trained with 1 million and 20 million words. We also report the best possible accuracy after spelling correction, where we assume that a word can be corrected after spelling correction, if it has a correct version in the candidate list. There are two reasons why a word can not be corrected:

1. The mis-spelled version of that word is another word in the language. In this case, since the spelling checker does not consider it as a mis-spelled word, it does not return any correction candidate.



Test File	Best Possible Accuracy	Language Model	Training Data Size	Accuracy
1	98.01%	Word-based	1 million words	95.37%
			20 million words	95.88%
		Prefix-Suffix	1 million words	94.93%
			20 million words	95.29%
2	97.94%	Word-based	1 million words	95.59%
			20 million words	96.10%
		Prefix-Suffix	1 million words	95.22%
			20 million words	95.44%
3	96.76%	Word-based	1 million words	93.13%
			20 million words	94.26%
		Prefix-Suffix	1 million words	91.99%
			20 million words	92.94%
4	96.39%	Word-based	1 million words	92.79%
			20 million words	93.90%
		Prefix-Suffix	1 million words	91.40%
			20 million words	92.21%

Table 7.3: The accuracy results for the spelling correction.

2. The spelling corrector could not find and return the correct version in the candidate set. This may occur either because the correct word is not in the language, that is, the correct word may be a foreign word, or because the word is not in the database of the spelling checker, that is, it may be a rare proper name.

When we use more training data, the accuracy of the spelling corrector increases as expected. However, for spelling correction, we couldn't achieve better results with the prefix-suffix language models. The word-based language models outperformed the prefix-suffix language models in all our experiments.

## 7.5 Discussion

Unlike the  $n$ -best list rescoring task, we could not achieve better results using the prefix-suffix language models. This has several reasons. For example, if the error is outside the prefix and suffix region (i.e., the 5th letter in a 10 letter word is substituted with another letter), and one of the candidates has a probable suffix and prefix, but the whole candidate is very rare, this wrong candidate is selected by the prefix-suffix model, whereas the word-based model discards it.

The spelling corrector sometimes outputs very rare words as candidates, because it has a very large vocabulary. Also, the surface forms of the candidates are very similar to each other, since the spelling checker looks for words that have an edit-distance of at most 2 letters from the mis-spelled word. However, the speech recognizer has a vocabulary size of 10,000 words. Therefore, the surface forms of the candidates are not very similar. The prefix-suffix models benefit from this property.

The speech recognizer outputs an ordered list of candidates for each word, that is, the lists at the top of the list are more probable than the ones at the bottom. The recognizer uses the acoustic signals in assigning the candidate words a probability. However, the spelling checker outputs only the candidate words, and does not assign any probability to each candidate, using information such as the confusability of the erroneous parts. A spelling corrector may benefit a lot from the usage of such information.

# Chapter 8

## Conclusions and Future Work

### 8.1 Summary

We have presented our approaches to statistical modeling for agglutinative languages, such as Turkish, especially those having productive derivational phenomena.

Our approach for morphological disambiguation essentially involves breaking up the full morphological analysis across derivational boundaries and treating the components, which we call as inflectional groups, as subtags, and then determining the correct sequence of subtags via statistical techniques. In order to model the distribution of inflectional groups we used various methods based on the dependency relationships. We tried both  $n$ -gram language modeling approach, and maximum entropy modeling approach for estimating the probabilities in our models.  $n$ -gram language models gave better accuracy results for morphological disambiguation, therefore we continued using this approach in our further task.

For  $n$ -best list rescoring for speech recognition and spelling correction, we have approximated IG-based language modeling approach using prefix-suffix language models. We compared word-based, IG-based and prefix-suffix language models for reducing the word error rate of a speech recognizer. We obtained the best

accuracy using prefix-suffix language models for  $n$ -best list rescoring. The reason behind this is that, we do not need labeled data to train these models, so we trained these models using much more data. Also, these models do not introduce ambiguity, so they do not introduce additional search space. We also analyzed the effect of root and suffix length for best accuracy. However, for spelling correction, the prefix-suffix models did not give the best performance. The best results were achieved using the word-based models, although there is very little difference in accuracy. Similar to the IG-based models, since the spelling checker proposes a large number of candidates for each word, the search space increases a lot. Also, since the prefix-suffix models only look at the beginning and end of the words, any mistake out of the scope of the prefix and suffix parts remains unresolved.

This, to our knowledge, is the first detailed attempt in statistical modeling of agglutinative languages and can certainly be applied to other such languages like Hungarian and Finnish with productive derivational morphology or other Turkic languages. Similar techniques can also be used in other applications like syntactic parsing, morphological analysis. Note that, we can benefit from the use of a statistical morphological analyzer, which returns probabilities along with the parses.

In the following subsections, we summarize the contributions of this thesis, and then present our suggestions/ideas for future work.

## 8.2 Contributions

The contributions of this thesis can be grouped under three subtopics.

### 8.2.1 Theoretical Contributions

This thesis presents a pioneering effort for statistical language modeling of Turkish. Previous statistical natural language processing studies have used words as

an appropriate unit for language modeling, which is suitable for languages like English. For languages with richer morphology, the best unit for language modeling might be smaller than a word, like morphemes or other units such as inflectional groups. This thesis searched for the best unit for modeling Turkish. We built statistical models in order to effectively use these smaller units, and used them successfully in our tasks. We found that, using inflectional groups instead of the whole parse sequences for morphological disambiguation, and prefix and suffixes instead of the whole words for  $n$ -best list rescoring and spelling correction, gave very good results. While building these systems, we computed the vocabulary size and the perplexity of Turkish, using a large corpus.

### 8.2.2 Experimental Contributions

While searching for the most appropriate unit for modeling Turkish, we performed various experiments with different techniques. We also reported results using different amounts of training data, and with variations of our techniques. For example, for morphological disambiguation using  $n$ -gram modeling, we built three different models, modifying our assumptions, or for speech recognition, we compared word-based, IG-based, and prefix-suffix language models. However, note that this study would be more complete if we had a large vocabulary *continuous* speech recognizer system which uses language models during recognition.

### 8.2.3 Contributions for Further Studies on Turkish

For training our word-based models, we collected 20 million word corpus from the web archives of a Turkish daily newspaper, using a web robot, which filters unnecessary information, such as HTML tags. Further data can be collected using this robot, on other web sites. This robot is publicly available for research purposes.

We morphologically analyzed and preprocessed (in order to reduce ambiguity) a 1 million word portion of this corpus. The preprocessor module is also publicly

available. We then disambiguated this portion, using our best model. Both the ambiguous and disambiguated versions of this corpus is available at:

<http://www.nlp.cs.bilkent.edu.tr/Center/Corpus/>

During this thesis work, we have developed a morphological disambiguation system and language models for Turkish, these are also available for further studies on Turkish.

## 8.3 Future Work

We have obtained satisfactory results for both tasks, using our language models. However, there is still room for further improvement. Clustering and real stem and suffix models can be investigated as a future work. Automatic acquisition of language modeling units is also promising.

### 8.3.1 Real Stem-Suffix Models

Instead of using a pre-defined length for extracting the prefix and suffix part of the words, we can stem all the words in the  $n$ -best list, and construct two models for the stem and the suffix part. For example, the word ‘buzdolabında’ (‘in the refrigerator’ in English) can be divided into two parts as follows:

buzdolabında → buzdolabı-nda

Here, the root and suffix parts are separated with a ‘-’. However, there will again be ambiguity for words like ‘koyun’ or ‘kadın’, for which multiple segmentations are possible:

koyun → koy-un OR koyu-n OR koyun

kadın → kadı-n OR kadın

We can also use a statistical stemmer, which returns probabilities for possible stem-suffix segmentations, and incorporate these probabilities into the corresponding application.

### 8.3.2 Class-based Models

We can cluster the space of all our modeling units in order to reduce the data sparseness problem. For example, we can use:

- words for word-based modeling,
- IGs for IG-based modeling,
- prefixes and/or suffixes for prefix-suffix modeling, and
- stems and/or suffixes for real stem-suffix modeling

For clustering, any of the automatic clustering techniques (that we cite in Chapter 2) can be used.

It is also possible to use some rules based on heuristics for clustering. For example, using the fact that the surface forms of the Turkish suffixes change as a result of the vowel harmony, the suffixes (both for prefix-suffix models and for real stem-suffix models) can be clustered according to their consonants and vowels. For example, both of the following suffixes used for genitive case marking of nouns:

nin, nın, nun, nün, in, ın, un, ün

can be clustered and modeled as a single suffix.

### 8.3.3 Automatic Acquisition of Language Modeling Units

In this study, we used our intuition and linguistic knowledge about Turkish to select the language modeling units. However, there are also methods for selecting appropriate units for language modeling [Kieczya *et al.*, 1999], which start from letters and then combine letter sequences in order to obtain a better language modeling unit. Methods like this one can also be investigated as a future work.



# Bibliography

- [Bahl *et al.*, 1983] Bahl, Lalit R.; Jelinek, Frederick; and Mercer, Robert L. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5(2):260–269.
- [Bangalore, 1996] Bangalore, Srinivas 1996. "almost parsing" techniques for language modeling. In *Proceedings of the 4<sup>th</sup> International Conference on Spoken Language Processing (ICSLP-96)*. 1169–1172.
- [Berger *et al.*, 1996] Berger, Adam; Pietra, Stephen Della; and Pietra, Vincent Della 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1).
- [Beyerlein *et al.*, 1998] Beyerlein, Peter; Aubert, Xavier; Haeb-Umbach, Reinhold; Klakow, Dietrich; Ullrich, Meinhard; Wendemuth, Andreas; and Wilcox, Patricia 1998. Automatic transcription of English Broadcast News. In *Proceedings of the 1998 DARPA Broadcast News Transcription and Understanding Workshop*, Landsowne, Virginia.
- [Birtürk, 1998] Birtürk, Ayşenur 1998. *A Computational Analysis of Turkish Using the Government-Binding Approach*. Ph.D. Dissertation. Middle East Technical University, Department of Computer Engineering.
- [Borthwick *et al.*, 1998] Borthwick, Andrew; Sterling, John; Agichtein, Eugene; and Grishman, Ralph 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In Charniak, Eugene, editor 1998.

- Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, Quebec, Canada. 152-160.
- [Brants, 2000] Brants, Thorsten 2000. TnT—a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*. Seattle, WA.
- [Brill, 1995a] Brill, Eric 1995a. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21(4):543-566.
- [Brill, 1995b] Brill, Eric 1995b. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third Workshop on Very Large Corpora*. Cambridge, MA.
- [Brown *et al.*, 1992a] Brown, Peter F.; Della Pietra, Stephen A.; Della Pietra, Vincent J.; Lai, Jenifer C.; and Mercer, Robert L. 1992a. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics* 19(1):31-40.
- [Brown *et al.*, 1992b] Brown, Peter F.; Pietra, Vincent J. Della; DeSouza, Peter V.; Lai, Jennifer C.; and Mercer, Robert L. 1992b. Class-based *n*-gram models of language. *Computational Linguistics* 18(4):467-480.
- [Çarkı *et al.*, 2000] Çarkı, Kenan; Geutner, Petra; and Schultz, Tanja 2000. Turkish LVC'SR: towards better speech recognition for agglutinative languages. In *ICASSP 2000: IEEE International Conference on Acoustics, Speech and Signal Processing*. İstanbul, Turkey.
- [Çiğdem Keyder Turhan, 1997] Keyder Turhan, Çiğdem 1997. An English to Turkish machine translation system using structural mapping. In *Proceedings of the Fifth Conference on Applied NLP (ANLP)*. Washington, D.C.
- [Charniak *et al.*, 1993] Charniak, Eugene; Hendrickson, Curtis; Jacobson, Neil; and Perkowski, Mike 1993. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Menlo Park, CA. AAAI Press/MIT Press. 784-789.

- [Charniak, 1993] Charniak, Eugene 1993. *Statistical Language Learning*. The MIT Press.
- [Chen and Goodman, 1996] Chen, Stanley and Goodman, Joshua 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, Santa Cruz, California. 310-318.
- [Church, 1988] Church, Kenneth W. 1988. A stochastic parts program and a noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas.
- [Cover and Thomas, 1991] Cover, Thomas M. and Thomas, Joy A. 1991. *Elements of Information Theory*. John Wiley and Sons, Inc. chapter 15. 474-478.
- [Şehitoğlu, 1996] Şehitoğlu, Onur 1996. A sign-based phrase structure grammar for turkish. Master's thesis, Middle East Technical University, Department of Computer Engineering.
- [Cutting *et al.*, 1992] Cutting, Doug; Kupiec, Julian; Pedersen, Jan; and Sibun, Penelope 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy.
- [Darroch and Ratcliff, 1972] Darroch, J.N. and Ratcliff, D. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics* 43:1470-1480.
- [Davenport *et al.*, 1999] Davenport, Jason; Nguyen, Long; Matsoukas, Spyros; Schwartz, Richard; and Makhoul, John 1999. The 1998 BBN BYBLOS 10x real time system. In *Proceedings of the DARPA Broadcast News Workshop*, Herndon, Virginia.
- [Dermatas and Kokkinakis, 1995] Dermatas, Evangelos and Kokkinakis, George 1995. Automatic stochastic tagging of natural language texts. *Computational Linguistics* 21(2):137-163.
- [DeRose, 1988] DeRose, S. J. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics* 14:31-39.

- [Elworthy, 1995] Elworthy, David 1995. Tagset design and inflected languages. In *From Texts to Tags: Issues in Multilingual Language Analysis, Proceedings of the ACL SIGDAT Workshop*, University College, Belfield, Dublin, Ireland. 1-9.
- [Erguvanli, 1979] Erguvanli, Emine Eser 1979. *The Function of Word Order in Turkish*. Ph.D. Dissertation, University of California, Los Angeles.
- [Ezeiza *et al.*, 1998] Ezeiza, N.; Alegria, I.; Arriola, J. M.; Urizar, R.; and Aduriz, I. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 36<sup>th</sup> Annual Meeting of the Association for Computational Linguistics and 17<sup>th</sup> International Conference on Computational Linguistics*, Montreal, Quebec, Canada. 379-384.
- [Fang and Huckvale, 2000] Fang, A. C. and Huckvale, M. 2000. Enhanced language modelling with phonologically constrained morphological analysis. In *ICASSP 2000: IEEE International Conference on Acoustics, Speech and Signal Processing*, Istanbul, Turkey.
- [Gale, 1994] Gale, William A. 1994. Good-Turing smoothing without tears. Technical report, Bell Labs. Can be found at <http://cm.bell-labs.com/cm/ms/departments/sia/doc/94.5.ps>.
- [Garside, 1988] Garside, Roger 1988. *The Computational analysis of English : a corpus-based approach*. Longman, London. chapter The CLAWS word-tagging system. 30-41.
- [Güngördü and Oflazer, 1995] Güngördü, Zelal and Oflazer, Kemal 1995. Parsing Turkish using the lexical-functional grammar formalism. *Machine Translation* 10(4).
- [Hajić and Hladká, 1998] Hajić, Jan and Hladká, Barbora 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36<sup>th</sup> Annual Meeting of the Association for Computational Linguistics (COLING/ACL'98)*, Montreal, Canada. 483-490.

- [Hakkani *et al.*, 1998] Hakkani, Dilek Zeynep; Tür, Gökhan; Mitamura, Teruko; Eric H. Nyberg, 3rd; and Oflazer, Kemal 1998. Prototype Interlingua-Based MT System from English to Turkish. In *Proceedings of Association for Machine Translation in the Americas (AMTA-98)*, Langhorne, PA. Also in the Springer Verlag Lecture Notes for Computer Science series.
- [Hakkani-Tür *et al.*, 2000] Hakkani-Tür, Dilek Z.; Oflazer, Kemal; and Tür, Gökhan 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18<sup>th</sup> International Conference on Computational Linguistics (COLING-2000)*.
- [Hankamer, 1989] Hankamer, Jorge 1989. *Lexical Representation and Process*. The MIT Press. chapter Morphological Parsing and the Lexicon.
- [Heeman and Allen, 1997] Heeman, Peter A. and Allen, James F. 1997. Incorporating pos tagging into language modeling. In *Proceedings of the 5th European Conference On Speech Communication and Technology (EUROSPEECH)*. Rhodes, Greece. 2767–2770.
- [Jaynes, 1957] Jaynes, E. T. 1957. Information theory and statistical mechanics. *Physics Reviews* 106 620–630.
- [Jelinek, 1998] Jelinek, Frederick 1998. *Statistical Methods for Speech Recognition*. The MIT Press.
- [Jurafsky and Martin, 1999] Jurafsky, Daniel and Martin, James H. 1999. *Speech and Language Processing*. Prentice Hall.
- [Karlsson *et al.*, 1995] Karlsson, Fred; Voutilainen, Arto; Heikkilä, Juha; and Anttila, Arto 1995. *Constraint Grammar-A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.
- [Katz, 1987] Katz, 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume assp-35:3. 400–401.
- [Kiecza *et al.*, 1999] Kiecza, Daniel; Schultz, Tanja; and Waibel, Alex 1999. Data-driven determination of appropriate dictionary units for Korean LVCSR.

- In *Proceedings of the International Conference on Speech Processing (ICSP'99)*, Korea.
- [Kwon, 2000] Kwon, Oh-Wook 2000. Performance of LVCSR with morpheme-based and syllable-based recognition units. In *ICASSP 2000: IEEE International Conference on Acoustics, Speech and Signal Processing*, Istanbul, Turkey.
- [Levinger *et al.*, 1995] Levinger, Moshe; Ornan, Uzzi; and Itai, Alon 1995. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics* 21(3):383–404.
- [Mangu and Brill, 1997] Mangu, Lidia and Brill, Eric 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the International Conference on Machine Learning*. 734–741.
- [Manning and Schütze, 1999] Manning, Christopher D. and Schütze, Hinrich 1999. *Foundations of Statistical Natural Processing*. The MIT Press.
- [Martin *et al.*, 1995] Martin, Sven; Liermann, Jörg; and Ney, Hermann 1995. Algorithms for bigram and trigram word clustering. In *Proceedings of European Conference on Speech Communication and Technology*, Madrid, Spain. 1253–1256.
- [McMahon and Smith, 1996] McMahon, John and Smith, Francis J. 1996. Improving statistical language model performance with automatically generated word hierarchies. *Computational Linguistics* 22:217–247.
- [Megyesi, 1999] Megyesi, Beáta 1999. Improving Brill's POS tagger for an agglutinative language. In Fung, Pascale and Zhou, Joe, editors 1999, *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Maryland, USA. 275–284.
- [Merialdo, 1994] Merialdo, Bernard 1994. Tagging English text with a probabilistic model. *Computational Linguistics* 20(2):155–172.

- [Mikheev *et al.*, 1999] Mikheev, Andrei; Moens, Marc; and Grover, Claire 1999. Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL '99)*. Bergen, Norway. 1-8.
- [Mikheev, 1998] Mikheev, Andrei 1998. Feature lattices for maximum entropy modelling. In *Proceedings of the 36<sup>th</sup> Annual Meeting of the Association for Computational Linguistics (COLING-ACL '98)*. Montreal, Canada. 848-855.
- [Niesler and Woodland, 1996] Niesler, T.R. and Woodland, P.C. 1996. A variable-length category-based n-gram language model. In *Proceedings of the International Conference on Audio, Speech, and Signal Processing (ICASSP)*. 164-167.
- [Oflazer and Güzey, 1994] Oflazer, Kemal and Güzey, Cemalettin 1994. Spelling correction in agglutinative languages. In *Proceedings of the 4<sup>th</sup> ACL Conference on Applied Natural Language Processing*, Stuttgart, Germany.
- [Oflazer and Kuruöz, 1994] Oflazer, Kemal and Kuruöz, İlker 1994. Tagging and morphological disambiguation of Turkish text. In *Proceedings of the 4<sup>th</sup> Applied Natural Language Processing Conference*. ACL. 144-149.
- [Oflazer and Tür, 1996] Oflazer, Kemal and Tür, Gökhan 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In Brill, Eric and Church, Kenneth, editors 1996, *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*.
- [Oflazer and Tür, 1997] Oflazer, Kemal and Tür, Gökhan 1997. Morphological disambiguation by voting constraints. In *Proceedings of the 35<sup>th</sup> Annual Meeting of the Association for Computational Linguistics (ACL '97/EACL '97)*. Madrid, Spain.
- [Oflazer *et al.*, 1999] Oflazer, Kemal; Hakkani-Tür, Dilek Z.; and Tür, Gökhan 1999. Design for a Turkish treebank. In *Proceedings of Workshop on Linguistically Interpreted Corpora, at EACL '99*. Bergen, Norway.

- [Oflazer, 1994] Oflazer, Kemal 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing* 9(2):137-148.
- [Oflazer, 1996] Oflazer, Kemal 1996. Error-tolerant finite state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics* 22(1).
- [Oflazer, 1999] Oflazer, Kemal 1999. Dependency parsing with an extended finite state approach. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland.
- [Pictra *et al.*, 1997] Pietra, Stephen Della; Pietra, Vincent Della; and Lafferty, John 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(4):380-393.
- [Rabiner, 1989] Rabiner, Lawrence R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257-286.
- [Ratnaparkhi, 1996] Ratnaparkhi, Adwait 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania.
- [Ratnaparkhi, 1998a] Ratnaparkhi, Adwait 1998a. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. Dissertation, University of Pennsylvania.
- [Ratnaparkhi, 1998b] Ratnaparkhi, Adwait 1998b. Unsupervised statistical models for prepositional phrase attachment. In *Proceedings of the 17<sup>th</sup> International Conference on Computational Linguistics*, Montreal, Canada.
- [Reynar and Ratnaparkhi, 1997] Reynar, Jeffrey C. and Ratnaparkhi, Adwait 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5<sup>th</sup> ACL Conference on Applied Natural Language Processing (ANLP'97)*, Washington, D.C.



- [Rosenfeld, 1996] Rosenfeld, Ronald 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language* 10:187–228.
- [Sankar *et al.*, 1998] Sankar, Ananth; Weng, Fuliang; Rivlin, Ze'ev; Stolcke, Andreas; and Gadde, Ramana Rao 1998. Development of SRI's 1997 Broadcast News transcription system. In *Proceedings of the 1998 DARPA Broadcast News Transcription and Understanding Workshop*. Landsowne, Virginia. 91–96.
- [Shannon, 1948] Shannon, Claude E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27:379–423 and 623–656.
- [Solak and Oflazer, 1993] Solak, Aysin and Oflazer, Kemal 1993. Design and implementation of a spelling checker for turkish. *Literary and Linguistic Computing* 8(3).
- [Stolcke, 1999] Stolcke, Andreas 1999. SRILM—the SRI language modeling toolkit. <http://www.speech.sri.com/projects/srilm/>.
- [Tür, 1996] Tür, Gökhan 1996. Using multiple sources of information for constraint-based morphological disambiguation. Master's thesis, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey.
- [Tür, 2000] Tür, Gökhan 2000. *A Statistical Information Extraction System For Turkish*. Ph.D. Dissertation, Department of Computer Engineering, Bilkent University, Ankara, Turkey.
- [Viterbi, 1967] Viterbi, Andrew J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13(2):260–269.
- [Voutilainen, 1998] Voutilainen, Aro 1998. Does tagging help parsing? A case study on finite state parsing. In Karttunen, Lauri and Oflazer, Kemal, editors 1998, *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing (FSMNLP'98)*, Bilkent University, Ankara, Turkey. 25–36.

- [Wegmann *et al.*, 1999] Wegmann, Steven; Zhan, Puming; Carp, Ira; Newman, Michael; Yamron, Jonathan P.; and Gillick, Larry 1999. Dragon systems' 1998 Broadcast News transcription system. In *Proceedings of the DARPA Broadcast News Workshop*. Herndon, Virginia.
- [Woodland *et al.*, 1999] Woodland, P.C.; Hain, T.; Moore, G.L.; Niesler, T.R.; Povey, D.; Tuerk, Andreas; and Whittaker, E.W.D. 1999. The 1998 HTK Broadcast News transcription system: Development and results. In *Proceedings of the DARPA Broadcast News Workshop*. Herndon, Virginia.
- [Yilmaz, 1999] Yilmaz, Cemal 1999. A large vocabulary speech recognition system for Turkish. Master's thesis, Computer Engineering Department, Bilkent University.
- [Young, 1995] Young, Steve 1995. Large vocabulary continuous speech recognition: A review. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, Snowbird, Utah. 3-28.

# Appendix A

## Categories

In the maximum entropy modeling approach, we represent each inflectional using 9 categories. The values that these categories may take are listed in the following sections.

### A.1 Major Part-of-Speech

Adj	Adjective
Adv	Adverb
BDTag	Begin Document Tag
BSTag	Begin Sentence Tag
BTTag	Begin Title Tag
Conj	Conjunction
Det	Determiner
Dup	Duplication
EDTag	End Document Tag
ESTag	End Sentence Tag
ETTag	End Title Tag

Interj	Interjunction
Noun	Noun
Num	Number
Postp	Postposition
Pron	Pronoun
Punc	Punctuation
Ques	Question
Verb	Verb

"

## A.2 Minor Part-of-Speech

Able  
Agt  
ByDoingSo  
Card  
Caus  
DemonsP  
Distrib .  
FitFor  
FutPart  
Inf  
Ness  
PersP  
Ord  
Pass  
PastPart  
Percent  
PresPart  
Prop  
QuantP  
QuesP

Range  
Ratio  
Real  
Recip  
Reflex  
ReflexP  
Time  
When  
While  
With  
Without  
WithoutHavingDoneSo  
Zero  
PCAbl  
PCAcc  
PCDat  
PCGen  
PCIns  
PCNom  
Acquire  
ActOf  
AfterDoingSo  
Almost  
As  
AsIf  
Become  
EverSince  
FeelLike  
Hastily  
InBetween  
JustLike  
Ly  
NotState  
Related

Repeat  
 Since  
 SinceDoingSo  
 Start  
 Stay

### A.3 Agreement

A1pl 1st person plural  
 A1sg 1st person singular  
 A2pl 2nd person plural  
 A2sg 2nd person singular  
 A3pl 3rd person plural  
 A3sg 3rd person singular  
 - No agreement

### A.4 Possessive

P1pl 1st person plural  
 P1sg 1st person singular  
 P2pl 2nd person plural  
 P2sg 2nd person singular  
 P3pl 3rd person plural  
 P3sg 3rd person singular  
 Pnon No possessive marker  
 - No possessive category

## A.5 Case

Abl	Ablative
Acc	Accusative
Dat	Dative
Equ	Equative
Gen	Genitive
Loc	Locative
Ins	Instrumental
Nom	Nominative
-	No case category

## A.6 Polarity

Pos	Positive
Neg	Negative
-	No polarity category

## A.7 Tense-Aspect-Mood Marker 1

Aor	Aorist
Desr	Desire
Fut	Future
Imp	Imperative
Narr1	Narrative
Neces	Necessity
Opt	Optative
Past1	Past
Pres	Present
Prog1	Progressive 1
Prog2	Progressive 2
-	No TAM1 marker

## A.8 Tense-Aspect-Mood Marker 2

- Cond Conditional
- Narr2 Narrative
- Past2 Past
- No TAM2 marker

## A.9 Copula

- Cop Copula
- No copula or no copula category