PREFIX-SUFFIX BASED STATISTICAL LANGUAGE MODELS OF TURKISH

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Umut Topkara
July, 2001

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

Asst. Prof. Dr. İlyas Çiçekli (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 thesis for the degree of Master of Science.

Asst. Prof. Dr. Uğur Doğrusöz

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

Asst. Prof. Dr. David Davenport

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

PREFIX-SUFFIX BASED STATISTICAL LANGUAGE MODELS OF TURKISH

Umut Topkara
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. İlyas Çiçekli

July, 2001

As large amount of online text became available, concisely representing quantitative information about language and doing inference on this information for natural language applications have become an attractive research area. Statistical language models try to estimate the unknown probability distribution $\hat{P}(u)$ that is assumed to have produced large text corpora of linguistic units u. This probability distribution estimate is used to improve the performance of many natural language processing applications including speech recognition (ASR), optical character recognition (OCR), spelling and grammar correction, machine translation and document classification. Statistical language modeling has been successfully applied to English. However, this good performance of approaches to statistical modeling of English does not apply to Turkish. Turkish has a productive agglutinative morphology, that is, it's possible to derive thousands of word forms from a given root word through adding suffixes. When, statistical modeling by word units is used, this lucrative vocabulary structure causes data sparseness problems in general and serious space problems in time-memory critical applications such as speech recognition.

According to a recent Ph.D. thesis by Hakkani-Tür, using fixed size prefix and suffix parts of words for statistical modeling of Turkish performs better than using whole words for the task of selecting the most likely sequence of words from a list of candidate words emitted by a speech recognizer. After these successful results, we have made further research on using smaller units for statistical modeling of Turkish. We have used fixed number of syllables for prefix and suffix parts. In our experiments we have used small vocabulary of prefixes and suffixes to test the robustness of our approach. We also compared the performance of prefix-suffix language models having 2-word context with word 2-gram models. We have found a language model that uses subword units and can perform as well as a large word

based language model in 2-word context and still be half in size.

Keywords: Statistical Language Modeling, Natural Language Processing, Agglutinative Languages, Speech Recognition, N-best List Rescoring, n-gram Language Models, Prefix Suffix Language Models.

ÖZET

TÜRKÇE'NİN ÖNEK-SONEK TABANLI İSTATİSTİKSEL MODELLERİ

Umut Topkara
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. İlyas Çiçekli
Temmuz, 2001

Teknolojik gelişmelerle beraber büyük derlemlerin ortaya çıkmasından sonra dil hakkındaki nicel bilgilerin özlü bir halde saklanması ve bu bilgi üzerinde çıkarımlar yapılması çekici bir bilimsel araştırma alanı haline geldi. İstatistiksel dil modelleri u dil birimlerinden oluşan büyük derlemleri ürettiği varsayılan ve bilinmeyen bir $\hat{P}(u)$ olasılık dağılımını tahmin etmekte kullanılırlar. Bulunan bu olasılık dağılımı tahmini , aralarında konuşma tanıma(speech recognition), yazım ve gramer hatalarını düzeltme, otomatik belge tercümesi ve otomatik belge sınıflandırmanın da bulunduğu birçok doğal dil işleme uygulamasının başarımını artırmak için kullanılabilir. İstatistiksel dil modelleme, İngilizce'ye başarıyla uygulanmıştır, ancak istatistiksel modellerin bu başarısı Türkçe'nin istatistiksel modellerine Türkçe'nin belirli özelliklerinden dolayı yeterince yansımamaktadır. Türkçe üretken sondan eklemeli bir dil yapısına sahiptir, yani bir kelime kökünden arka arkaya eklemeler yoluyla binlerce kelime üretmek mümkün olmaktadır. Kelime birimleri üzerinden istatistiksel modeller kullanıldığında Türkçe'nin üretken sözlük yapısı genel olarak veri yetersizliğine ve konuşma tanıma gibi zaman-yer kritik uygulamalarda ciddi yer ve zaman problemleri oluşturmaktadır.

Yakın zamanda tamamlanan Hakkani-Tür'e ait doktora tezindeki bulgulara göre, Türkçe için konuşma tanıma uygulamalarının ürettiği aday listelerinin yeniden değerlendirilmesinde, kelimelerin sabit büyüklükteki önek ve sonek birimleri üzerinden yapılan n-birimli istatiksel modeller kelime birimleri üzerinden yapılan n-birimli modellere göre daha iyi başarı sağlamaktadırlar. Bu başarılı sonuçlardan sonra, kelimeden küçük birimler üzerinden Türkçe'nin istatistiksel modelleri konusunda daha fazla araştırma yaptık. Çalışmalarımızda önek

ve sonek kısımları için sabit sayıda hece kullanılan çeşitli istatistiksel modeller denedik. Yaklaşımlarımızın güçlülüğünü değerlendirebilmek için önek ve sonek dağarcığımızı kısıtlı tuttuk.

Ayrıca 2 kelime birimi bağlamlı önek sonek modellerimizin başarımını kelime birimleri üzerinde 2-birimli istatistiksel modellerle karşılaştırdık. Araştırmalarımızın sonunda 2 kelime bağlamda kelime tabanlı dil modeliyle aynı performansı gösteren, ancak yarı boyutta olan bir dil modeli geliştirdik.

Anahtar sözcükler: İstatistiksel Dil Modelleme, Doğal Dil İşleme, Sondan Eklemeli Diller, Konuşma Tanıma, Aday Listesi Değerlendirme, n-birimli Dil Modelleri, Önek Sonek Dil Modelleri.

Acknowledgement

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One theory (Jerison, 1991) is that human language stems primarily from a need for better cognitive maps of the territory. Canines and other social carnivores rely heavily on scent marking and their olfactory system to decide both where they are and what other animals have been there... But the early hominid (monkeys and apes 30 million years ago) did not have a well enough developed olfactory system to map out the world this way, so they substituted vocal sounds for scent marking. Thus, the rest of this chapter, in Jerison's view, is devoted to the way we humans compensate for our inadequate noses.

Stuart Russell and Peter Norvig

The Evolution of Language, Artificial Intelligence A Modern Approach[27], page 653

1.1 Overview

Statistical language modeling is the process of building an estimate for the probability distribution of natural language units, which vary from words to sentences and documents[26]. This probability distribution estimate is used to improve the performance of many natural language processing applications including speech recognition (ASR), optical character recognition (OCR), spelling and grammar correction, machine translation and document classification.

Statistical language modeling has been successfully applied to languages like English [15]. Currently there is ongoing research to apply the existing speech technology to other languages like Arabic, Dutch, German, Japanese, Spanish, Mandarin Chinese, Korean and statistical modeling of these languages is an important constituent of this effort. Although the acoustic models have been found to be transferable[28] between languages due to the small phone repertoire of humans, this hasn't been the case for language models. Language models of some languages, including the Turkish language, need extra care for their special characteristics. In Turkish language, a word may have different realizations in the sentence depending on the part of speech role it plays. Thus, some preprocessing is required to successfully apply traditional approaches to Turkish.

Experiments in a recent thesis by Hakkani-Tür[10] have shown that statistical language modeling by dividing a word into a fixed size prefix and a fixed size suffix achieves competitive improvements on the language model performance for the task of selecting the most likely sequence of words from a list of candidate words emitted by a speech recognizer, when compared to more expensive language modeling methods.

This thesis presents our attempts to overcome some problems of statistical language modeling of Turkish due to its highly productive agglutinative morphology by using prefix-suffix n-gram models. We have tested various prefix suffix models in 2 and 3-word context and one of these models can perform as good as a word-based model but is half in size. The research in this thesis is mainly built up on and projected from the findings of PhD thesis by Hakkani-Tür[10].

1.2 Motivation

Turkish language has a rich agglutinative morphology, which has productive inflectional and derivational suffixations. The order of the constituents in Turkish may change freely in sentences according to the discourse context. Such characteristics make Turkish language more complicated to process with statistical approaches[10, 37] when compared with languages like English, which has less productive morphology and fixed constituent order. The free constituent ordering in Turkish introduces irregularity and its rich morphology introduces a huge number of distinct tokens as complexities to be tackled.

Morphological constituents of words have been successfully used for reducing the out of vocabulary rate and word error rate[9, 3]. However due to inherent ambiguity in Turkish, the existing morphological analyzer requires about 10% of the analyzed words to be further processed by human experts[11]. Moreover the morphological analysis step raises a computational overhead that may be unwanted. Our research has been towards finding prefix and suffix portions of words that will reduce the vocabulary size and at the same time represent words' morphological identity that is significant to preserve the word error rate of the speech recognizer.

1.3 Layout of the Thesis

The focus of this thesis is statistical language modeling, therefore in the next chapter we elaborate on statistical language modeling with emphasis on n-gram language models. There is also a short review of speech recognition in this chapter. A brief information about characteristics of the Turkish language is given in the beginning of Chapter 3, followed by a discussion of problems with its statistical modeling. As a solution to these problematic issues of Turkish, implications of using various prefix-suffix language models are presented. Also in the same chapter, application of statistical language modeling on speech recognition is explained. After this we describe our statistical models for Turkish in detail. In the fourth chapter results of various prefix-suffix language modeling approaches to improve speech recognition performance is reported. Finally in the fifth chapter we have listed our conclusions from this work.

Chapter 2

Statistical Language Modeling

Information theory has perhaps ballooned to an importance beyond its actual accomplishments.
Claude Elwood Shannon

To the man who only has a hammer in the toolkit, every problem looks like a nail.
Abraham Maslow

In this chapter we introduce the notion of statistical language modeling with emphasis on n-gram language models. After we give methods for deriving statistical language models from corpora and improving these derived language models, we explain a powerful computational tool used for statistical learning called Hidden Markov Models. We conclude this chapter with a brief introduction to speech recognition, from the viewpoint of an application domain for statistical language models.

2.1 Statistical Language Modeling

As large amount of online text became available, concisely representing quantitative information about language and doing inference on this information for natural language applications became an attractive research area. Statistical language models try to estimate the unknown probability distribution $\hat{P}(u)$ that is assumed to have generated large text corpora of linguistic units u. The estimated distribution $P(u)$ expresses distribution of all possible u that can ever exist in the language.

The first successful application of statistical language modeling was for integrating language information to optical character recognition in a statistical framework. Currently speech recognition is the most widely used application of language models. In speech recognition, the aim is to find the sequence of words W^* that has the highest likelihood $P(W|A)$, given an acoustic waveform A produced by an utterance $\hat{W}$, as in Equation 2.1[1]. Ideally $W^* = \hat{W}$.

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

Since it's not practically possible to directly compute $P(W|A)$, Bayes' rule is used to decompose it to computable probabilities as in Equation 2.2. $P(W)$ is the probability distribution of word sequence W and $P(A|W)$ is the acoustic likelihood. Since $P(A)$ is independent of W, it's just a normalization constant to keep the result as a legal probability and may be ignored in computation of W^*.

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

Statistical language models(SLMs) are used to compute $P(W)$ values in Equation 2.2. N-gram models, part of speech based models, structured models and exponential models are some of statistical modeling techniques that are used to find $P(W)$[26]. Details of various statistical language modeling techniques can be found in books on statistical language processing[19, 15].

Almost all SLMs represent the probability of a sentence W as:

[1] *argmax* function returns the argument that maximizes the expression on its right side.

$$P(W) \stackrel{def}{=} P(w_1^n) = \prod_{i=1}^{n} P(w_i|h_i) \tag{2.3}$$

where $W = w_1^n$ is[2] a sentence of n words and $h_i = w_1^{i-1}$ is the history of i^{th} word. As can be seen in Equation 2.3 the problem of computing the probability of a sentence W is very similar to predicting the next word from previous words:

$$P(w_i|h_i)$$

The task of predicting the next word from its history is referred to as the Shannon Game after his seminal paper[31] in information theory that used such prediction experiments to present his findings.

The word prediction task is actually a classification task[19], in which classificatory features(i.e. past words) are used to predict a target feature(i.e. next word). In order to do this prediction, we make some independence assumptions, that is, we leave some classificatory features unconsidered; either because the target feature does not depend on them or the dependency is minor that we can neglect them to be able to afford computational costs. Effectively we divide the data into certain equivalence classes that have the same values for a certain subset of the significant classifiers, and these equivalence classes are used in the prediction. The number of classificatory features that we take into consideration in prediction task determines the accuracy and reliability of our predictions to some extend. More classificatory features means a more accurate classification, because every small effect acting on the target feature is considered. Whereas this makes the prediction unreliable due to the empirical characteristic of the task, which will produce less instances of the equivalence classes in the training data. However, it is not possible to get better results indefinitely as more data is made available. The performance reaches a limit after some point due to reasons that will be discussed at the end of this section.

Before elaborating more on statistical language modeling techniques, we will discuss evaluation criteria for language models in the next section. Section 2.1.2

[2] From now on we will use u_i^j and $u_i, u_{i+1}, \ldots, uj$ alternatively to denote the sequence of units from subscript i to j

discusses n-gram language models, followed by two variations of it: part-of-speech based n-gram models and class based n-gram models. We continue with a subsection on estimating language model parameters and then discuss advantages and disadvantages of statistical natural language modeling.

2.1.1 Performance of Statistical Language Models

2.1.1.1 Word Error Rate

As language models are just aides to NLP applications, their performance is best measured in terms of the benefit they provide to the application's performance. The word error rate (WER) of the application in which a language model is integrated is a metric of this kind. WER is rate of the total number of deletions, insertions and substitutions found by computing the minimum edit distance between the hypothesized sentence (*hypothesis*) by the recognizer and the real utterance (*reference*), to the total number of words in the real utterance.

$$WER = 100 \times \frac{insertions + deletions + substitutions}{number\ of\ words\ in\ \hat{W}}$$

However, most of the time speech recognizer programs are not readily available, and such performance measures are hard to gather, hence WER is not appropriate for comparing the performance of different language models. For this reason evaluation metrics that can be calculated efficiently and without the need for a speech recognizer are most commonly used. Among these common metrics are *entropy*, *cross-entropy* and *perplexity*.

In this thesis, since we had speech recognizer data available to us, we have evaluated the quality of our models in terms of WER.

2.1.1.2 Entropy

Entropy is the amount of information[3] that is captured in a random variable measured in bits. It is the statistical measure of the number of bits needed to encode a message in language L when the most efficient coding is used from the view of *communication theory* [24]. For a random variable X, which varies over the linguistic units u that our language model is trying to predict, let $P(x)$ be the probability distribution of X. Then the entropy $H(X)$ of this random variable is:

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

2.1.1.3 Cross-entropy

The main issue in statistical modeling is to construct a framework that captures the underlying process and to feed this framework with training data in a way that it gathers as much information about the underlying processes as possible. For this reason, the likelihood of the unseen data D with respect to the language model M is also used as a performance measure. The average log likelihood of data $D = (D_1, D_2, \ldots, D_n)$ with respect to model M gives this quantity in bits:

$$AverageLogLikelihood(D|M) = \frac{1}{n} \sum_{d \in D} log P_M(d) \tag{2.4}$$

Average Log Likelihood $(D|M)$ is an approximation to the cross entropy $H(P, P_M)$ of true data distribution P that generated D with respect to model distribution P_M:

$$H(P, P_M) = -\frac{1}{n} \sum_{d \in D} P(d) log P_M(d) \tag{2.5}$$

[3] The term *information* is used in the context of information theory.

Cross-entropy $H(P, P_M)$ is provably greater than the real entropy of language $H(P)$. So it may be used as an upper bound approximation to it.

2.1.1.4 Perplexity

Most of the time *perplexity* is reported as a performance measure in SLM research.

$$perplexity(P, P_M) = 2^{H(P,P_M)} \tag{2.6}$$

The perplexity value of a language model on a data D having value k means that on the average you are surprised as if you had to guess between k choices that have the same probability.

Although perplexity is a common evaluation metric in SLM research community, it does not directly translate to an improvement in performance of the target application. It is even possible for models with same perplexity to have about 1% word error rate difference[34]. It's worth mentioning that, in [26] a 5% reduction in perplexity is regarded as insignificant and 10%-20% reduction as sometimes translating to increase in recognition rate.

2.1.2 n-gram Language Models

In the task of predicting the next word, the history of word w_n is the classificatory feature set that is used to predict target feature w_n. A possible independence assumption is to assume that the last few words determine the next word, which is also called the Markov Assumption. A 3-gram language model assumes dependence of a word only to previous two words:

$$P(w_i|h_i) \approx P(w_i|w_{i-2}, w_{i-1})$$

The probability $P(w_i|w_{i-2}, w_{i-1})$ can be computed from:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{\sum_w C(w_{i-2}, w_{i-1}, w)}$$

and by setting $C(w_i, ..., w_n)$ to the frequency of word sequence $w_i, ..., w_n$. The n-gram frequencies returned by function $C(.)$ are derived from a *training corpus* that is a representative of the actual domain which we are going to apply our language models to.

The performance of the language models are reported as their performance on a *test corpus* that hasn't been seen before by the language models. Sometimes another test corpus may be needed to adjust parameters before actually using the models, such corpora are called *dev-test corpora or held-out corpora*.

Since sum of all 3-gram counts that start with a particular word subsequence w_{i-2}, w_{i-1} is equal to the 2-gram counts $C(w_{i-2}, w_{i-1})$, the formula above can be rewritten as:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

The general formula for estimating n-gram parameters uses *relative frequency*:

$$P(w_i|w_{i-n+1}, ..., w_{i-1}) = \frac{w_{i-n+1}, ..., C(w_i)}{w_{i-n+1}, ..., C(w_{i-1})} \tag{2.7}$$

This parameter estimation technique is called *Maximum Likelihood Estimation(MLE)*. Note that the probabilities calculated using Equation 2.7 maximizes the probability of the training set it is derived from.

Because the number of parameters to be estimated will be quite large, it is usually not practical to use more than 3-grams. There are approximately 1.773×10^{10} possible 3-grams that can be constructed with a vocabulary consisting of 260,740 different words. An English corpus of 365,893,263 words contains only 75,349,888 of these possible 3-grams, and only 8,728,789 of these 3-grams occur more than three times in the corpus[2].

According to *Zipf's Law*, the frequency f of a word type is related to rank r of the word as $f \propto \frac{1}{r}$. For this reason, in practice, only n-grams for words that occur over a given frequency threshold is computed. All other words are regarded as *out-of-vocabulary(OOV)* items and are represented with <**unk**>. This strategy does not ruin the performance of the system, but reduces the parameter space drastically when *hapax legomena* , words that occurred once in training corpus, are excluded from n-gram computations. This is due to the fact that hapax legomena usually constitute half of the word types, but they count for only a small portion of tokens, conforming to Zipf's Law.

2.1.3 POS-based and Class-based Models

In this section we are going to present POS [4]-based and class-based Models that are used to overcome the large parameter space size problem in language modeling. As mentioned in the beginning of Section 2.1, a large number of features makes it hard to train the parameters of a statistical language model. Assume that V_{type} is the number of word types in language. In the case of n-grams, we need to train $O(V_{type}^n)$ parameters[5], and even with very large corpora it is not practical to use more than 3-gram word models. However it's possible to replace a classificatory feature set with another feature set that has a similar predictive power, yet less sparse. A common practice for English is to use stemming which will reduce the number of parameters by replacing the vocabulary of word types with the smaller *vocabulary of stems.*

Another approach to the sparseness problem is to use the *part of speech(POS)*of the words for computing the posterior probability of w_i[14]. Let POS_i denote the POS tag of the i^{th} word, then:

$$\begin{aligned} P(w_i|w_{i-n+1}^{i-1}) &= P(w_i, POS_i|w_{i-n+1}^{i-1}, POS_{i-n+1}^{i-1}) \\ &= P(w_i|w_{i-n+1}^{i-1}, POS_{i-n+1}^{i})P(POS_i|w_{i-n+1}^{i-1}, POS_{i-n+1}^{i-1}) \end{aligned}$$

A simplification of this model might be:

$$P(w_i|w_{i-n+1}^{i-1}) = P(w_i|POS_i)P(POS_i|POS_{i-n+1}^{i-1})$$

The POS based approach reduces the parameter space complexity of the n-gram language models drastically. An improvement over POS based language models are class based language models. In class based language models clustering algorithms[2] that use information theoretic clues are used to find an n-to-1

[4]POS stands for Part Of Speech that a word takes in a particular sentence.
[5]In fact exact value of independent variables is $V^n - 1$[2].

function π that maps a word w_i to a word class c_i. A class-based language model computes the posterior probability of w_i given the succeeding string of words, for $1 \leq i \leq n$, as:

$$P(w_i|w_1^{i-1}) = P(w_i|c_i)P(c_i|c_1^{i-1}) \tag{2.8}$$

In class-based language models the main question is to effectively and efficiently assign classes to each individual word in the vocabulary. There are several clustering algorithms that divide vocabularies into classes using metrics such as mutual information[2] as described in Section 2.1.4. A large reduction in the complexity over a word-based n-gram language model can be achieved by using a class-based n-gram language model, while keeping the performance of the language model at the same level. When a clustering algorithm [2] is applied on the corpus mentioned in previous section, to yield 1000 classes, among the 1.000×10^9 possible class 3-grams only 26,913,330 actually occurred in the corpus, and 6,917,746 of these 3-grams occurred more than three times. The number of observed 3-grams increase by a factor of 2 in this class-based model, moreover applying interpolated estimation to this class-based model has made a slight improvement over the perplexity of the word-based interpolated language model of the same corpus. In the next subsection we are goind to introduce the clustering algorithm used in this class-based model.

2.1.4 Mutual Information Maximization Based Clustering

Brown et.al.[2] have shown that maximizing 2-gram class model parameters is equivalent to maximizing the mutual information between consecutive classes $I(c_i, c_j)$, in a training corpus, where:

$$\begin{aligned} I(c_i, c_j) &= H(c_i) - H(c_i|c_j) \\ &= H(c_j) - H(c_j|c_i) \end{aligned}$$

According to Brown et.al. there is no practical algorithm that can guarantee such clustering, but their greedy suboptimal algorithm still yields some interesting classes. The algorithm has three main steps. In the first step word 2-gram probabilities in the corpus are obtained and mutual information between each ordered class pair is calculated from these 2-gram probabilities. After data preparation is completed in the first step, the next two steps are iteratively applied until C classes remain. Each class pair is checked for the amount of decrease in the mutual information if the pair of classes are merged, and class pair yielding minimum decrease is selected for merging. Then the 2-gram probabilities of classes and mutual information among classes are updated to reflect the merge between the classes. Each serial processing of update and search for the best merge are done in parallel requiring $O(V^2)$ operations for each merge.

After $V-k$ merges, let $q_k(l,m)$ be mutual information between classes l and m and $p_k(l,m)$ be the probability of the 2-gram (l,m). Then,

$q_k(l,m) = p_k(l,m) log \frac{p_k(l,m)}{\sum_z p_k(l,z) \sum_z p_k(z,m)}$, and

$p_k(i+j,m) = p_k(i,m) + p_k(j,m)$, and

$q_k(i+j,m) = p_k(i+j,m) log \frac{p_k(i+j,m)}{\sum_z p_k(i+j,z) \sum_z p_k(z,m)}$.

Let $s_k(i)$ denote the sum of the mutual information of 2-grams that contain i as a component on any side of its associated ordered pair, then,

$s_i = \sum_l q_k(l,i) + \sum_m q_k(i,m) - q_k(i,i)$.

It is clear that, after k merges have been done, if classes i and j are merged new total mutual information $I_k(i,j)$ will be obtained by first subtracting the sum of mutual information calculations that involve any of classes i and j from the total mutual information in that step I_k and adding the mutual information of 2-grams with the new class $i+j$.

$$\begin{aligned} I_k(i,j) &= I_k - s_k(i) - s_k(j) + q_k(i,j) + q_k(j,i) \\ & \sum_{l \neq i,j} q_k(l,i+j) + \sum_{m \neq i,j} q_k(i+j,m) + q_k(i+j,i+j) \qquad (2.9) \end{aligned}$$

[1] initialize and find first $\{i, j\} \in maxpair$ that maximizes $I_k(i, j)$
[2] for k from V to $V - C$
[3]   merge words in classes $i, j \in maxpair$, i, j, into class i
[4]   if $j \neq k$ rename class k to j
[5]   set I_{max} to 0
[6]   for each class pair l, m, such that $l < m$
[7]   update $p_{k-1}(l, m)$, $q_{k-1}(l, m)$, $I_{k-1}(l, m)$
[8]   if $I_{k-1}(l, m) \geq I_{max}$
[9]   set I_{max} to $I_k(l, m)$
[10]   set $maxpair = \{l, m\}$
[11]   for each class $l \neq i$
[12]   update $I_{k-1}(l, i)$ and $I_{k-1}(i, l)$
[13]   update $s_{k-1}(l)$

As the *update* operations inside the *for* loop, which starts on line 11, take $O(V^2)$ operations the total running time of the algorithm is $O(V^3)$.

2.1.5 Smoothing

> *"When you have eliminated the impossible, that which remains, however improbable, must be the truth."*
>
> Sherlock Holmes, by Sir Arthur Conan Doyle (1859-1930), British writer, physician.

Even after using feature selection methods, some word sequences among V^n possible sequences will not be observed in the training data and will be given zero probability by our probability estimation method. Such estimation policy is sound for illegal word sequences. However, not all legal word sequences will occur in the training data and these sequences will be unfairly assigned zero probabilities, because the models will be *fit* to the training data. In order to compensate for this error in probability assignment, we should apply a procedure that will give none-zero probability to unseen events and decrease the probability of seen events, which is called *smoothing*.

2.1.5.1 Good-Turing Smoothing

In Good-Turing smoothing, n-grams that have low counts are estimated from the n-grams with higher counts. Let N_c be the number of n-grams with count c:

$$N_c = \sum_{W:C(W)=c} 1$$

The smoothed count c^* for the n-grams with count c is:

$$c^* = (c+1)\frac{E(N_{c+1})}{E(N_c)} \tag{2.10}$$

where E is the expectation of a random variable[5]. In Good-Turing smoothing $E(N_1)/N$ of the probability mass is distributed to unseen n-grams.

However, it is not possible to know $E(N_c)$, and empirical N_c might be substituted for it. Using the empirical values for expectation in formula 2.10 is somewhat problematic. For high values of c, estimates will be inaccurate due to the scarcity of such samples. In order to overcome this, one might either use Good-Turing estimation for n-gram frequencies less than a threshold k or we might fit the N_c versus c values to a function and use that function instead of empirical values.

2.1.5.2 Back-off Smoothing

It is also possible to use lower order n-gram counts to estimate n-gram probabilities. In *back off smoothing*[18] n-gram models of lower orders are recursively tried for a non-zero count and the n-gram count found is discounted to find P_D and interpolated to guarantee that total probability mass does not exceed 1.

$$P(w_i|w_{i-2}, w_{i-1}) = \begin{cases} P_D(w_i|w_{i-2}, w_{i-1}), & if C(w_{i-2}, w_{i-1}, w_i) > 0 \\ \alpha_1 P(w_i|w_{i-1}), & if C(w_{i-1}, w_i) > 0 \\ \alpha_2 P(w_i), & otherwise \end{cases} \tag{2.11}$$

It's possible to find the discounted probability P_D by using Good-Turing smoothing.

Besides back-off smoothing may be used to combine probabilities from different knowledge sources.

2.1.5.3 Deleted Interpolation

Deleted interpolation[16] is another method that combines lower order models to estimate a better n-gram probability. In deleted interpolation all n-gram models are linearly interpolated by weighting with λ linear weights :

$$\begin{aligned} P(w_i|w_{i-2}, w_{i-1}) &= \lambda_1 P(w_i|w_{i-2}, w_{i-1}) \\ &+\lambda_2 P(w_i|w_{i-1}) \\ &+\lambda_3 P(w_i) \end{aligned} \tag{2.12}$$

λ values are chosen so as to make $\sum \lambda_i = 1$, which ensures the total probability mass is 1. It's also possible to choose different λ values for different contexts, to emphasize accurate n-gram estimations. λ values can be chosen both by hand or by an algorithm called *Expectation Maximization (EM)* to maximize the probability assigned to a *held-out corpus.* A variant of EM will be discussed in Section 2.2.2.3.

2.1.6 Advantages and Disadvantages of Statistical Language Models

I have a spelling checker
It came with my PC;
It plainly marks four my revue
Mistakes I cannot sea.
I've run this poem threw it,
I'm sure your pleased too no,
Its letter perfect in it's weigh,
My checker tolled me sew.

Janet Minor

Statistical language models are known to improve the performance of their target applications very strongly. Statistical language models are easy to train without much human labor and models of different language phenomena may be easily combined in a uniform framework. However, common SLM techniques have some deficits. The most important of them is that the SLM are generally overfit to their target domain and their performance does not transfer to other domains although they're hard to beat in the domains for which they're built[26]. Also statistical language models use little or no linguistic information, and for this reason, after some point, their performance does not get better with increasing size of training text. A commonly shared view of SLM is that its future lies in incorporating linguistics to it[26].

2.2 Hidden Markov Models (HMMs)

> *As linguists, we find it a little hard to take seriously problems over an alphabet of four symbols, but bioinformatics is a well-funded domain to which you can apply your new skills in Hidden Markov Modeling!*
>
> C. Manning and H. Schütze,
>
> Foundations of Statistical Natural Language Processing[19], pp 338.

There are two main models that may be used to characterize the properties of a given output, deterministic models and statiscial models. Statistical models try to capture the characteristics of a real world process as a parametric random process, such that these parameters can be determined in a well defined way. This thesis is about modeling language from the statistical perspective, and Markov Processes and Hidden Markov Models are central to the ideas that are presented.

2.2.1 Markov Processes

Discrete Markov Models are a subclass of statistical models. Suppose we have a system that may be at any of the states from a finite state set $S = \{S_1, S_2, ...S_N\}$. At regularly spread discrete times $t = 1, 2, ...$ the system changes state or stays at the same state, such that q_t is the state at time t. A Markov Model can be viewed as a stochastic process whose output is the current state of a finite state automaton with probabilities attached to state transition arcs. Figure 2.2.1 shows an example for such processes.

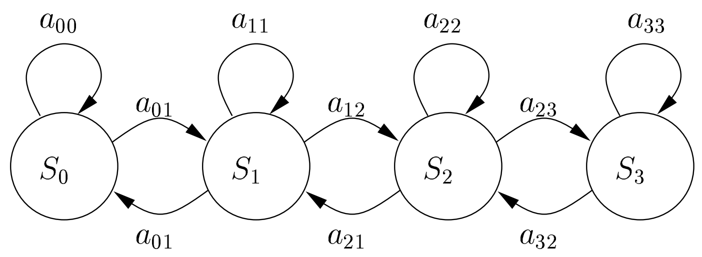


Figure 2.1: A Markov Chain with four states.

In a *first order discrete Markov Process (Chain)* M, the following *Markov Properties* hold:

- Limited Horizon: The state at time t is conditionally independent on the states at times $1, 2, ...t-2$ given the state at time $t-1$

$$P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, ...) = P(q_t = S_j | q_{t-1} = S_i)$$

- Time Invariant: The state transition probabilities are independent on time.

$$P(q_t = S_j | q_{t-1} = S_i) = a_{ij}, 1 \leq i, j \leq N$$
$$a_{ij} \geq 0, \sum_{j=1}^{N} a_{ij} = 1$$

Since the states that the machine visits in time are observable in this process, such processes are termed *visible(observable) Markov Models.*

Let $\Pi = \{\pi_0, \pi_1, ...\}$ be the initial state probabilities such that $\pi_i = P(q_1 = S_i)$. And let $O = \{o_1, o_2, ..., o_T\}$ be an output sequence of states, such that observation at time t, o_t, is the state observed at time t and $o_t \in S$. Then the probability of observing O can be easily calculated:

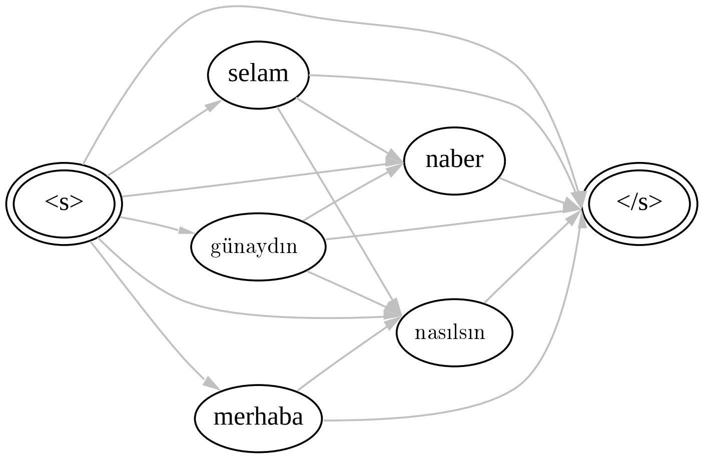
$$\begin{aligned} P(O|M) &= P(o_1, o_2, ...o_T) \\ &= P(o_1)P(o_2|o_1)P(o_3|o_2)...P(o_T|o_{T-1}) \\ &= \pi_{o_1} a_{o_1 o_2} a_{o_2 o_3} ... a_{o_{T-1} o_T} \end{aligned}$$

N-gram language models are in fact Markov Models. A 2-gram model is a Markov Chain where nodes correspond to words in the vocabulary and arcs carry transition probability identical to word 2-gram probabilities of the language model. Since n-grams with $n \geq 3$ look early to the history, it may seem as if such n-grams violate limited horizon and are not Markov Models. However it is possible to transform an n-gram language model with $n \geq 3$ to a visible markov model by storing $(n-1)$grams in the states. When a fixed finite amount of history m is stored in the states of a visible Markov Model, it is called m^{th} order Markov Model. See Figure 2.2 for 2-gram and 3-gram models corresponding to a sublanguage consisting of Turkish greetings that one hears in the mornings in Bilkent. Note that the state space of the Markov Model increases exponentially, as does the parameter space size of language models with increasing history(or context) size.

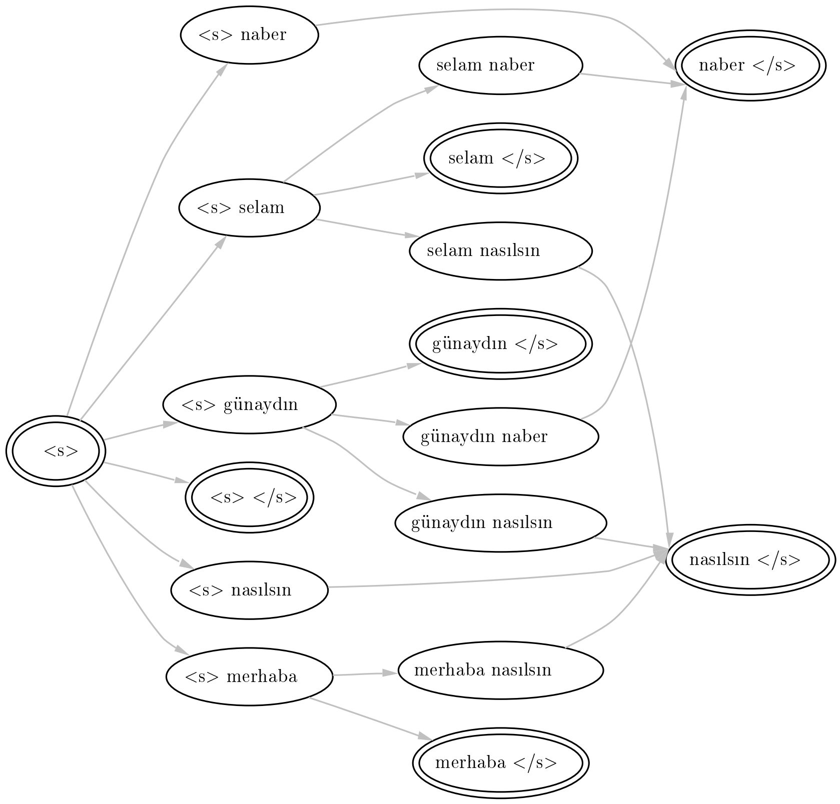
2.2.2 Hidden Markov Models

Visible Markov Models are of limited use to model real life processes. Hidden Markov Models(HMM) are an extension to visible Markov models in which outputs are observed as a probabilistic function of states. This probability of observing a certain output at a certain state is called *emission probability*. In other words, HMMs have two stochastic processes, one is the observed output emissions and the other is the hidden state transitions. In a HMM, the observations are statistically independent of each other and the states are correlated with the Markov property.

In the communications literature, the terms *Markov sources* or *probabilistic functions of Markov sources* is used instead of HMM. HMMs were first described by Baum and his colleagues in late 1960s.



a)



b)

Figure 2.2: State diagrams corresponding to 2-gram and 3-gram language models of different greetings you may hear in the mornings. Only non-zero probability arcs are shown and the probabilities on the arcs are not drawn.

The elements of HMMs are as follows:

- Observation symbols $V = \{v_1, \dots, v_M\}$
- Set of states $S = \{s_1, \dots, s_N\}$
- Observation sequence $O = \{o_1, \dots, o_T\}, \quad o_t \in V$
- State sequence $Q = \{q_1, \dots, q_{T+1}\}, \quad q_t \in S$
- Initial state probabilities $\Pi = \{\pi_i : 1 \leq i \leq N\}$
- State transition probabilities

$$A = \{a_{ij} : a_{ij} = P(q_{t+1} = s_j | q_t = s_i)\}$$

- Observation emission probabilities

$$B = \{b_{ij}(k) : b_{ij}(k) = P(o_t = v_k | q_t = s_i, q_{t+1} = s_j)$$

This type of HMM is called and *arc emission HMM*[19], since the outputs o_t are emitted during transition between states q_t and q_{t+1}. When the outputs o_t are emitted at state q_t the model is called a *state emission HMM*.

In order to specify a HMM λ, we need parameters A, B and Π to be defined.

$$\lambda = (A, B, \Pi)$$

We can easily simulate the behavior of a HMM with the following code segment:

$t = 1$
set $q_1 = s_i$ with probability π_i
while(true)
    make transition from s_i to s_j with probability a_{ij}
    emit symbol v_k with probability $b_{ij}(k)$
    $t = t + 1$

However such simulation is usually not interesting for our purpose in using HMMs. There are three main problems that are of concern about HMMs given their elements:

I. Given an observation sequence O and a model $\lambda = (A, B, \Pi)$, what is the probability that λ emits O?

II. Given an observation sequence O and a model $\lambda = (A, B, \Pi)$, what is the state sequence Q that best explains the observations?

III. How do we maximize the model parameters $\lambda = (A, B, \Pi)$ that maximizes the likelihood of observation O?

The following subsections present well-known answers to each of these questions[25].

2.2.2.1 Finding likelihood of an observation (forward algorithm)

The forward procedure is an efficient algorithm to compute the likelihood of an observation sequence given a Hidden Markov Model. Let $\alpha_t(i) = P(o_1, o_2, \ldots, o_{t-1}, q_t = s_i|\lambda)$, denote the probability of being in state s_i at time t after observing $o_1, \ldots, o_{t-1}$. It's possible to efficiently compute $P(O|\lambda)$ by memoizing α values. The algorithm is as follows:

I. Initialization:

$$\alpha_1(i) = \pi_i, 1 \leq i \leq N$$

II. Induction:

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_{ij}(o_t), \quad 1 \leq t \leq T, 1 \leq j \leq N$$

III. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_{T+1}(i)$$

It's also possible to use the dual of the forward procedure to compute $P(O|\lambda)$, i.e. by memoizing $\beta_t(i) = P(o_t, \ldots, o_T, q_t = s_i|\lambda)$:

I. Initialization:

$$\beta_{T+1}(i) = 1, 1 \leq i \leq N$$

II. Induction:

$$\beta_t(j) = \sum_{i=1}^{N} \beta_{t+1}(i) a_{ji} b_{ji}(o_t), \quad 1 \leq t \leq T, 1 \leq j \leq N$$

III. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i)$$

A more general form of the above formulations uses both forward and backward variables:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i), \quad 1 \leq t \leq T+1$$

2.2.2.2 Finding the best state sequence(Viterbi algorithm)

In order to find the optimal state sequence Q that maximizes $P(Q, O|\lambda)$, let us define the probability of the most likely path that ends at state q_t, at time t, after observing $o_1, \ldots, o_{t-1}$:

$$\delta_t(i) = \max_{q_1, \ldots, q_{t-1}} P(q_1, \ldots, q_t = s_i, o_1 \ldots o_{t-1}|\lambda)$$

and let $\psi_t(i)$ store the most likely previous state for s_i at time t. Viterbi algorithm computes the best state sequence in $O(|A|)$ time, where $|A|$ is the number of transitions in the automata representing the HMM, and *max* is a function that returns the maximum value of the expression on its right side:

I. Initialization:

$$\delta_1(i) = \pi_i, \quad 1 \leq i \leq N$$
$$\psi_1(i) = 0$$

II. Induction:

$$\delta_{t+1}(j) = \max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij} b_{ij}(o_t), \quad 2 \leq t \leq T$$
$$1 \leq j \leq N$$
$$\psi_t(j) = \operatorname*{argmax}_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij} b_{ij}(o_t), \quad 2 \leq t \leq T$$
$$1 \leq j \leq N$$

III. Termination:

$$P^* = \max_{1 \le i \le N} \delta_{T+1}(i)$$
$$Q^* = \operatorname*{argmax}_{1 \le i \le N} \delta_{T+1}(i)$$

2.2.2.3 Parameter estimation (Baum-Welch algorithm)

Although currently there is no algorithm to find the model parameters (A, B, π) that will maximize the likelihood of an observation sequence $P(O|\lambda)$, Baum and his colleagues[1] have found an iterative procedure to find a local maxima.

Let $\xi_t(i,j)$ denote the probability of being at state s_i at time t and s_j at time $t+1$, i.e. $\xi_t(i,j) = P(q_t = s_i, q_{t+1} = sj)$. $\xi_t(i,j)$ can be written in terms of forward and backward variables as:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_{ij}(o_t) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_{ij}(o_t) \beta_{t+1}(j)}$$

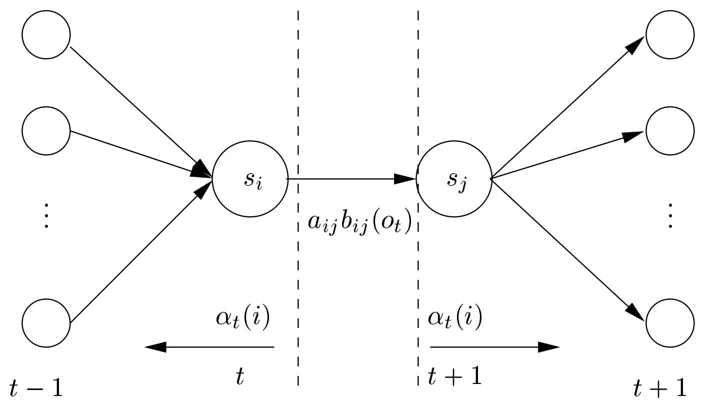


Figure 2.3: Probability of a transition from state s_i to s_j at time t given observation sequence O.

If we define $\gamma_t(i)$ as the probability of being in state i before observing o_t, then:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

Using γ and ξ values, it's possible to determine the expected number of transitions from the given state and the expected number of transitions between a given pair of states:

$$\sum_{t=1}^{T} \gamma_t(i) = expected\ number\ of\ transitions\ from\ s_i,\ given\ O$$

$$\sum_{t=1}^{T} \xi_t(i,j) = expected\ number\ of\ transitions\ from\ s_i\ to\ s_j,\ given\ O$$

If $\lambda = (A, B, \pi)$ is the initial model and $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ is the new estimation, the following stochastic constraints need to be preserved in each iteration:

$$\sum_{i=1}^{N} \bar{\pi} = 1$$

$$\sum_{j=1}^{N} \bar{a}_{ij} = 1, \quad 1 \leq i \leq N$$

$$\sum_{k=1}^{M} \bar{b}_{ij}(v_k) = 1, \quad 1 \leq i \leq N$$

The Baum-Welch algorithm iteratively modifies the model parameters so as to give more probability to transitions that are taken more while observing the *training data* O as follows:

$$\begin{aligned}
\bar{\pi}_i &= expected\ number\ of\ times\ q_1 = s_i \\
&= \gamma_1(i) \\
\bar{a}_{ij} &= \frac{expected\ number\ of\ transitions\ from\ s_i\ to\ s_j}{expected\ number\ of\ transitions\ from\ s_i} \\
&= \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \\
\bar{b}_{ij}(v_k) &= \frac{expected\ number\ of\ transitions\ from\ s_i\ to\ s_j\ emitting\ v_k}{expected\ number\ of\ transitions\ from\ s_i\ to\ s_j} \\
&= \frac{\sum_{\substack{t=1 \\ o_t=v_k}}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}
\end{aligned}$$

2.3 Speech recognition

Speech is one of the most natural forms of communication between humans. In the speech recognition task the aim is to find the sequence of words that were uttered to produce an acoustic signal sequence.

Russel and Norvig[27] decompose a communication between two agents S and H to seven episodes, where S wants to convey a proposition P to the hearer H using words W.

- Intention: S wants H to believe P.
- Generation: S chooses words W to express P.
- Synthesis: S utters words W as A.
- Perception: H hears $\hat{A}$ and perceives them as $\hat{W}$.
- Analysis: H infers $\hat{W}$ means one of $P_1, \ldots, P_n$.
- Disambiguation: H decides that S wanted to mean P_i with $\hat{W}$
- Incorporation: H decides to believe or reject P_i

The first three stages *intention, generation* and *synthesis* occur at the side of agent S, and the rest occur at H. Clearly a speech signal is used in the *synthesis* and *perception* stages of this decomposition, and *speech recognition* corresponds to the *perception* stage of the communication process that tries to perceive $\hat{W}$ aiming at $\hat{W} = W$. Knowledge of language is used from *perception* to *disambiguation* at the hearer side. Although speech communication is very natural and easy for humans, a speech recognizer that has the capabilities of a human has not been built yet.

Speech recognition is not the focus of this thesis so we are not going to elaborate on the details of a speech recognizer more than is needed to provide the context of the work we present in the big picture. The scientific literature on speech recognition is fairly rich and reader is referred to them for detailed information[19, 17].

Speech recognition can be characterized by the various constraints put on the task as follows[35]:

- Vocabulary Size: small(2-100), medium(100-1000), large(>1000).
- Speaking Style: read, spontaneous
- Language Domain: task oriented, unconstrained
- Speaker Dependence: speaker dependent, speaker independent
- Channel Quality: high bandwidth, low bandwidth
- Acoustic Environment: noisy, noiseless

Regardless of the type of the recognition task, state of the art speech recognizers work as statistical pattern recognizers[39]. The aim is to find the most likely word sequence $\hat{W}$ in language L given the observed acoustic signal A. Recall Equation 2.2 that decomposes the posterior probability of a word sequence to the product of prior probability of word sequence $P(W)$ and posterior probability of acoustic signal given word sequence $P(A|W)$. $P(W)$ is found by applying a language model as described in the previous sections. $P(A|W)$ is found by applying *acoustic models*.

Speech recognition systems works in several levels(Figure 2.4). At the *signal processing* level the acoustic observation is fragmented into overlapping frames and spectral features which possess energy information of these frames is then generated. HMM techniques are then used for *phone recognition* that assign $P(o|q)$, likelihoods of phones o, given the observation sequence. The HMMs used in speech recognition are state emission HMMs and each phoneme has its own HMM which usually has 3-states. The emitted symbols from the states are spectral feature vectors. The last stage is the *decoding* in which a *pronunciation dictionary* and a *language model* are used to decide the most likely hypotheses.

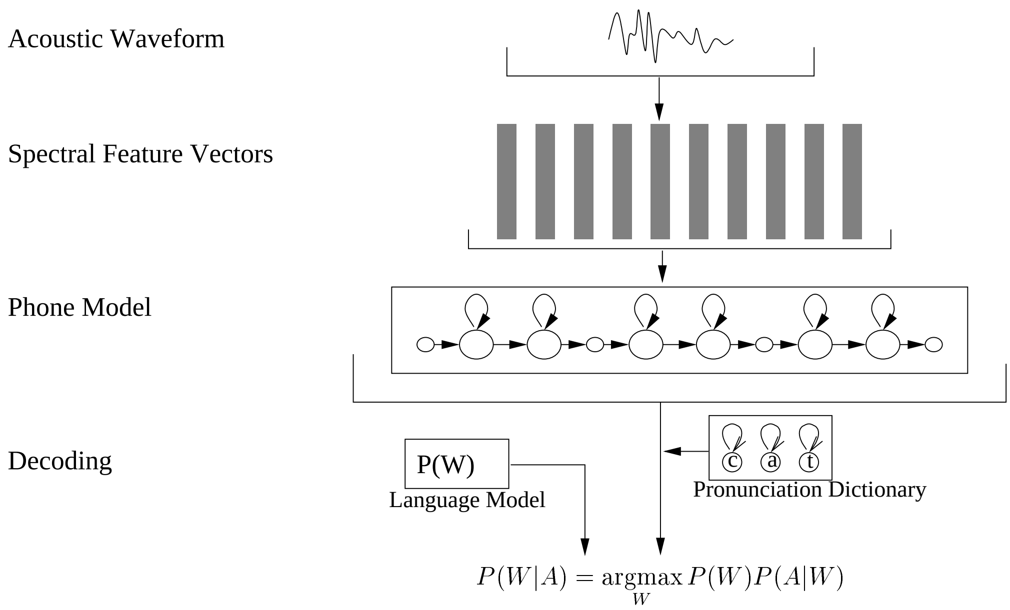


Figure 2.4: Components of a speech recognizer.

There are two main approaches to the decoding problem (i.e. to find the best $\hat{W}$): *depth-first* and *breadth-first*. *Viterbi decoding* is the main algorithm that breadth first strategies use. Since all possible hypotheses are searched in the Viterbi decoding, less expensive variations of it are used in practice, such as *beam search* which use only the highest probable words at each breadth first search level in order to seek the words in the next level. For ranking the hypotheses at this level, *language model lookeahead*[6] to lower n-gram models is used to integrate some language knowledge to this process. Depth-first strategies are A^* *search* and *stack decoding*.

Various limitations of the Viterbi decoder prevent it from being used as is in the decoding with complex language models. To fix these limitations a variation of the Viterbi algorithm that finds multiple best word sequences is used[30], and then expensive language models and acoustic models are applied on this best sequences. These best word sequences might be output as an *N-best list* consisting of top n sentence hypotheses with the highest scores or as a *word lattice*. The nodes in a word lattice correspond to recognized words and directed arcs between nodes encode a different utterance in each path from the start word to the end word. It's possible to encode the observation probability and the observation times to the nodes such that $P(W|A)$ can be computed using sophisticated language models.

The A^* decoding uses a priority queue to store the substrings of the hypotheses and iteratively finds the hypotheses that will augment the best substring using a *fast match* and pushes these new substrings back to the priority queue. The fast match is a class of efficient and effective heuristics that find possible words matching an acoustic sequence.

Chapter 3

Prefix-Suffix Models

"To the good listener, half a word is enough."

Spanish Proverb

"Once you've gotten the meaning, you can forget the words."

Chuang Tzu (BC 369- BC 286)

In this chapter we discuss the problems of statistical modeling of Turkish. We first start with a brief discussion of characteristics of the Turkish language with emphasis on its morphophonemic rules. After we introduce the previous work on modeling of Turkish, which we have built our study on, we discuss our approach. In the last section we introduce our application domain, on which we have experimented our proposed models.

3.1 Turkish

Turkish is a free word order language with agglutinative morphology. The Turkish language belongs to the Altaic language group.

In the Turkish language, word surface forms can be generated by adding suffixes to the root words. The suffixation can be of two types: *inflectional* and *derivational.* Inflectional suffixation augments part of speech information to a word, whereas derivational suffixation generates new words from the initial word with different meaning than the initial form. The morphology of Turkish is productive, that is, it's possible to generate millions of words[12] from a given Turkish root word. The morphotactics and morphophonemics of the Turkish language have been encoded in finite state transducers by Oflazer[21].

Constituents of sentences in the Turkish language may be placed in several positions according to discourse context. However the dominant word order is *subject object verb* . Since the morphology in Turkish assigns part of speech roles to words through inflectional suffixation, the place of these constituents in the sentence becomes insignificant for their part of speech role. A detailed study of the word order in Turkish has been done by Erguvanlı[7].

The morphophonemic rules of Turkish[33] determine how vowels and consonants change when a stem is followed by a suffix and such deformations generate variations of morphemes called allomorphs. Allomorphs of suffixes are significant for this study, because they increase the size of vocabulary of suffixes found in a given corpus. We will briefly discuss some of the main morphophonemic rules that generate allomorphs in the following subsections.

3.1.1 Vowel Harmony

Turkish has eight vowels *a, e, ı, i, o, ö, u, ü* which are classified according to the roundness, narrowness and place of tongue as in Table 3.1.1. According to the vowel harmony in Turkish, vowel of the succeeding syllables assimilate to the first vowel in frontness and roundness in a word with respect to the following rules:

1. Vowels assimilate to the preceding vowel in frontness.
2. A narrow vowel assimilates to the preceding vowel in rounding.
3. Among wide vowels, only unrounded wide vowels can occur in non-initial position.

In accordance with vowel harmony several variations of suffixes become possible. (*eve, ava, kola, söze*) and (*evi, avı, kolu, sözü*) show different allomorphs of dative suffix due to rule 1 and accusative suffix due to rule 2 respectively.

	Unrounded		Rounded	
	Wide	Narrow	Wide	Narrow
Back	a	ı	o	u
Front	e	i	ö	ü

Table 3.1: Vowels in the Turkish language.

3.1.2 Consonant Harmony

Consonant harmony in the Turkish language determines variations in final consonant of the stems and initial consonant of suffixes in affixation depending on the type of these consonants.

1. The voiced fricative consonants *b, c ,d, g* do not occur as the last letter of Turkish words, with some minor exceptions. Such voiced consonants in loan words are transformed into their voiceless counterparts *p, ç, t, k*, as in *harp* which is originally *harb* in Arabic.
2. In multisyllabic words and some monosyllabic words the voiceless stop consonants *p, ç, t, k* are transformed into *b, c, d, g, ğ*, as in dative form of *kitap* realized as *kitaba*.
3. *c, d, g* occuring in beginning of suffixes transform into *ç, t, k* when affixed to a word ending with a voiceless consonant *p, ç, t, k, f, h, s, ş*, as in variations of locative suffix *kitapta* and *evde*.
4. When a suffix that begins with a vowel is affixed to a word that ends with a vowel one of *y, s, ş, n* is inserted before the suffix, as in dative suffixation of *kapıyı* which contrasts with the dative examples in previous section.

- Sana kalemimi getirdim. (*)
- Sana getirdim kalemimi. (*)
- Kalemimi sana getirdim. (*)
- Kalemimi getirdim sana. (*)
- Getirdim kalemimi sana.
- Getirdim sana kalemimi.

Table 3.2: All word combinations of the sentence *"Sana kalemimi getirdim." (I brought my pencil to you)*, with the common ones marked with asterisk.

3.2 Language Modeling for Turkish

Extensive research has been carried out for achieving successful language models of the English language. Unfortunately, approaches to this problem do not fit perfectly to the Turkish language and similar languages. In the Turkish language, in contrast with the English language, constituents of a sentence might be ordered in different ways, reflecting different discourse contexts. Since the inflectional suffixation marks words with their part of speech role in the sentence, there is not a need for word ordering to determine part of speech. For example all six word combinations for sentence *Sana kalemimi getirdim (I brought my pencil to you)* in Table 3.2 are meaningful Turkish sentences, although only the ones marked with asterisk are common.

This freedom and irregularity in sentence structure cause serious complications for statistical language models, because such models aim to capture the regularities of languages they are applied to. For example, in order to train Turkish sentences that are like those in Table 3.2, we need four times larger training corpus than we would need for training similar ones in English.

Another complexity of languages like Turkish, from the viewpoint of statistical approaches, is the rich morphology that has agglutinative nature and productive derivational and inflectional suffixation. This lucrative morphology of Turkish is the reason for its huge set of distinct tokens. The size of a vocabulary in a corpus of 10M words can grow up to the order of 100,000 words in English, whereas in Turkish, vocabulary size can reach up to the order of 470,000 for a similar size of corpus[10]. In Figure 3.2, the rapid growth of vocabulary size in Turkish is contrasted to that of Italian, English and Finnish on small corpora of 800000 words, each of which were collected from news sites on the web on the same date.

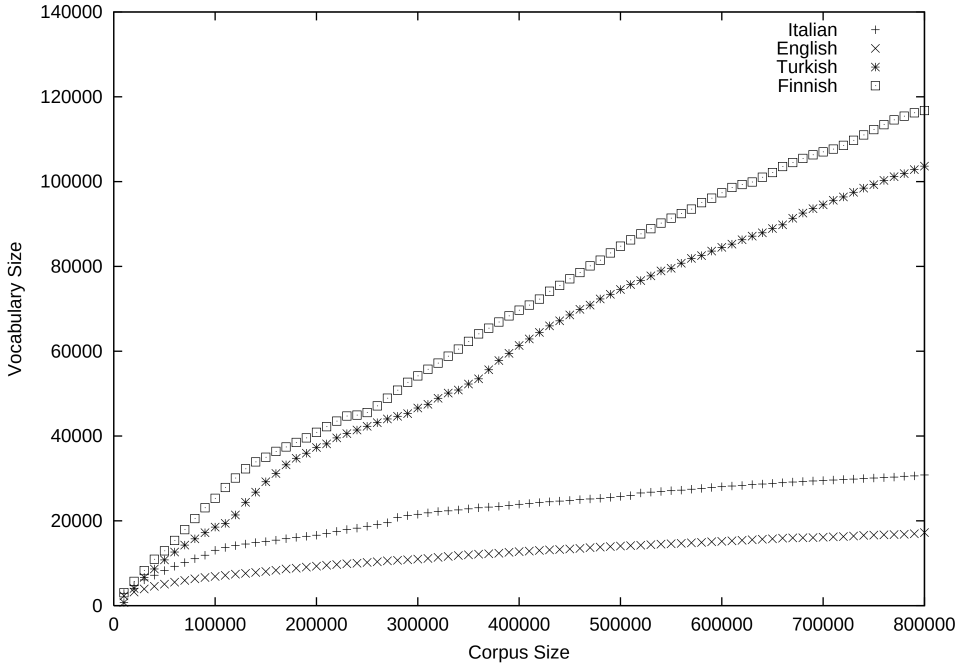


Figure 3.1: Comparison of vocabulary growth rates of English, Italian, Turkish and Finnish in news articles.

A language model that deals with Turkish also needs to deal with a huge number of units to model. This problem is reflected not only on the run-time complexity, but also on the accuracy of the language models, because of the data sparseness, as described in the previous chapter. Hakkani-Tür[10] has found perplexity of the Turkish language as approximately ten times greater than that of the English language(see Table 3.3). This gap between word-based statistical complexities of Turkish and English is due to integral impact of large vocabulary size and free constituent order of Turkish.

	Training Set Size	Perplexity
Turkish	10M words	1084.13
English	10M words	108.52

Table 3.3: The word-based perplexity values for 1M words of English and Turkish using a trigram language model.

3.3 Related Work

In order to overcome the difficulties introduced by large number of distinct tokens, processing the stem and suffix parts of the tokens as distinct units has produced successful results[10, 4].

Hakkani-Tür[10] has tried language modeling using information on inflectional groups, morphology, and prefix-suffix models. The prefix-suffix language models tried in that research has promising performance and this thesis reports further research on language modeling of Turkish with the prefix-suffix model approach.

3.3.1 Language Modeling Based On Inflectional Groups

Part of speech based statistical language models make use of language syntax for assigning a probability distribution to word sequences. The fact that data sparseness is less severe with part of speech based models makes them attractive to bypass problematic issues in language modeling of Turkish, because part of speech tags usually come from a small vocabulary of tags. However, tagging Turkish words turns out to be another problem. Languages like English might be tagged with a finite tag set, whereas in Turkish successive inflections of a word might possess different syntactic relationship information and a finite tagset is an inappropriate choice to cover all combinations of inflections. As a solution to this, Oflazer and his colleagues[23] have adapted viewing the part of speech information of Turkish words as a cumulative of all inflections and derivations appended to a root word. Figure 3.2 shows the morphological parse of the derived determiner *sağlamlaştırdığımızdaki*, which contains part of speech information.

```
sağlam+Adj^DB+Verb+Become^DB+Verb+Caus+Pos^DB+Adj+PastPart+P1sg^DB
+Noun+Zero+A3sg+Pnon+Loc^DB+Det
```

Figure 3.2: Morphological parse of the word *sağlamlaştırdığımızdaki*

An abstraction of the complex morphological syntax of Turkish is to to view suffixation of Turkish words as a sequence of inflectional groups (IG) separated by derivational boundaries:

$$\texttt{root} + \texttt{Infl}_1\texttt{\^{}DB} + \texttt{Infl}_2\texttt{\^{}DB} + \ldots\texttt{\^{}DB} + \texttt{Infl}_\texttt{n}$$

Figure 3.3 shows the list of IG's in the word *sağlamlaştırdığımızdaki.*

- `Adj`
- `Verb+Become`
- `Verb+Caus+Pos`
- `Adj+PastPart+P1sg`
- `Noun+Zero+A3sg+Pnon+Loc`
- `Det`

Figure 3.3: Inflectional groups of the word *sağlamlaştırdığımızdaki*

Empirical derivations on the inflectional groups imply interesting regularities that can be used for and exploited by statistical language modeling. Oflazer[22] has observed that in the majority of cases syntactic relationships only emanate from the last IG of the dependant word and land on one of the IG's of the word on the right. Moreover these syntactic relationships, if considered as directed links between nodes that represent words, do not cross, again with minor exceptions. This spatially regular nature of syntactic relationships between inflectional groups of words is exploited by IG-based statistical language models of Turkish, and prefix-suffix language models are variations of this model that also exploit this phenomenon. See Figure 3.4 for syntactic relations and part of speech roles of words in sentence "Bu eski bahçedeki gülün böyle büyümesi herkesi çok etkiledi."

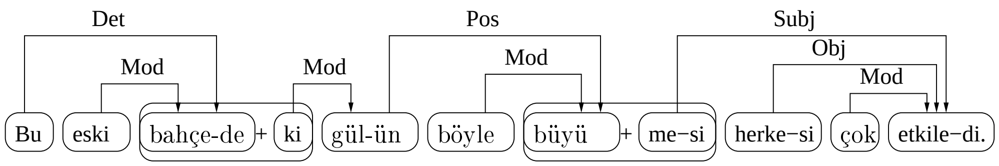


Figure 3.4: Example showing syntactic relationships of the sentence *"Bu eski bahçedeki gülün böyle büyümesi herkesi çok etkiledi."* represented with IG links.

In *inflectional group(IG) based language modeling*, just like POS-based language modeling[13], the probability of a sequence of words and their corresponding part of speech tags $P(W,T)$ is approximated with[10]:

$$P(W,T) = \prod_{i=1}^{n} P(w_i, t_i | w_1^{i-1}, t_1^{i-1}) \tag{3.1}$$

$$= \prod_{i=1}^{n} P(w_i | w_1^{i-1}, t_1^{i}) \times P(t_i | w_1^{i-1}, t_1^{i-1}) \tag{3.2}$$

$P(w_i|t_i) = 1$, when the root and the full morphological parse is used as the tag t_i for the word w_i, and the Equation 3.2 is reduced to:

$$P(W,T) = \prod_{i=1}^{n} P(t_i | t_1^{i-1}) \tag{3.3}$$

The morphological parses t_i for word w_i are generated by the morphological analyzer/generator developed by Oflazer [21]. After a sequence morphological parses is selected from candidate sequences, surface forms that correspond to this parse sequence are generated using the same morphological analyzer.

The morphological analysis and generation steps in the IG based language modeling are extra steps that should be integrated into the language model, thus they bring computational overhead and programming complexity to language modeling, which might be undesirable in some cases. Moreover, a significant fraction of Turkish words turn out to have inherent morphological ambiguity and when the morphological analyzer encounters such words it generates all possible parses for them. Since the morphological analyzer does not assign probabilities to such ambiguous parses, multiple parses that belong to a word are treated as if each have the same probability of being the real parse of that particular occurrence of the word, which may result in poor performance[10].

3.3.2 Prefix-Suffix Models

In order to eliminate the inconvenience of the morphological analysis and generation steps in the IG based model, Hakkani-Tür[10] has omitted these steps and used a fixed size of prefix and a fixed size of suffix extracted from the token, instead of the real root and IG of the word. In this model, the prefix and suffix pair of a word is treated as if they correspond to the root and last IG pair assigned to the word by the morphological analyzer. Thus the probability of a word sequence is approximated with:

$$\begin{aligned} P(W) &= \prod_{i=1}^{n} P(P_p(w_i)|P_p(w_1^{i-1})) \\ &\quad \times P(S_q(w_i)|S_q(w_1^{i-1})) \end{aligned} \tag{3.4}$$

where $P_p(w)$ is the initial p letters of word w, and $S_q(w)$ is the last q letters of word w. Applying the stem-suffix model not only tries to capture the morphology in agglutinative languages, but also decreases the number of distinct tokens to be processed by the language model. Hakkani-Tür has tried various values for (p, q) pair, namely $(3, 2)$, $(4, 2)$ and $(4, 3)$, the latter of which performs best.

3.4 New Prefix-Suffix Models

We have built a system to test various statistical prefix-suffix language models for Turkish. The approach in our models is similar to that of POS based language models and it takes the structure of the prefix-suffix model in Equation 3.4 as its foundation. Our aim is to find some simple language models that could improve over the quality of a word-based language model while being more compact when they are applied to the Turkish language.

In our new models, the ordered pair of prefix of length p, $P_p(w_i)$, and suffix $S_q(w_i)$ of length q of a word w_i, $S_q(w_i)$, is treated as the class of w_i, i.e.:

$$c(w_i) = (P_p(w_i), S_q(w_i)) \tag{3.5}$$

Until the end of this section, we will omit the lengths of prefix and suffix parts in our formulas for the sake of simplicity, i.e. we will refer to $P_p(w_i)$ and $S_q(w_i)$ as P_i and S_i.

The above class model transforms the probability of a word sequence W in Equation 3.4 into:

$$
\begin{aligned}
P(W) &= \prod_{i=1}^{n} P(w_i|P_i, S_i) \\
&\quad \times P(P_i, S_i|P_1^{i-1}, S_1^{i-1}) && (3.6)\\
&= \prod_{i=1}^{n} P(w_i|P_i, S_i) \\
&\quad \times P(P_i|P_1^{i-1}, S_1^{i-1}) \times P(S_i|P_1^{i}, S_1^{i-1}) && (3.7)\\
&= \prod_{i=1}^{n} P(w_i|P_i, S_i) \\
&\quad \times P(S_i|P_1^{i-1}, S_1^{i-1}) \times P(P_i|P_1^{i-1}, S_1^{i}) && (3.8)
\end{aligned}
$$

Note that the equivalence of Equation 3.6, Equation 3.7 and Equation 3.8 follows from the chain rule of probability theory (i.e. $P(A, B|C) = P(A|B, C) \times P(B|C)$).

In order to relieve the computational complexity raised by the large number of parameters in Equation 3.6 we make several simplifying assumptions.

Given a fixed size suffix and a fixed size prefix of a word, it is hard to compute the posterior probability $P(w_i|P_i, S_i)$ accurately for every possible word w_i. Finding this probabilty is equivalent to predicting the middle part M_i of the word that remains between prefix and suffix. Since the inflections in Turkish are productive there will be many alternatives to M_i that can fill the position for producing many different words. For this reason, rather than computing $P(w_i|P_i, S_i)$ empirically, we assume all such posterior probabilities to be equal for each word, and take this probability to be 1. So we have as our second simplification over word based model (the first was using a prefix-suffix method):

$$P(w_i|w_1^{i-1}) = P(P_i, S_i|P_1^{i-1}, S_1^{i-1}) \tag{3.9}$$

In this study we have limited our language models to 3-grams, so we approximate Equation 3.9 as:

$$P(w_i|w_1^{i-1}) = P(P_i, S_i|P_{i-2}^{i-1}, S_{i-2}^{i-1}) \tag{3.10}$$

In order to compute Equation 3.10 we should make some Markov assumptions[4] that encode the dependency of prefix and suffix of a word to prefixes and suffixes of previous words.

Model 1: The first method has already been mentioned. It assumes that the sequence of prefixes and sequence of suffixes embedded in a sequence of words are outputs of two independent random processes. The probability of a sequence of prefixes and suffixes in Equation 3.7 is now computed as in the following equation:

$$P(P_1^N, S_1^N) = \prod_{i=1}^{N} P(P_i|P_{i-2}^{i-1}) \times P(S_i|S_{i-2}^{i-1}) \tag{3.11}$$

Model 2: Our motivation in using suffixes and prefixes is the assumption that they capture some information of the part of speech and stems of words, and we can benefit from this hypothesis in a cost effective way. If such assumption is correct, then we should expect some correlation between the sequence of prefixes $P_1, P_2, \ldots, P_N$ and sequence of suffixes of a sentence $S_1, S_2, \ldots, S_N$ of N words.

$$P(P_i, S_i|P_1^{i-1}, S_1^{i-1}) = P(S_i|S_{i-2}^{i-1})P(P_i|S_{i-1}^{i}) \tag{3.12}$$

Recall that the above equation corresponds to a state emission HMM λ, where S_j, S_k are hidden state labels that are 2-grams of suffixes and P_l are emitted symbols from these states. The probability of a word sequence corresponds to $P(O, Q|\lambda)$, where O is the observation sequence(i.e. prefix sequence) and Q is the state sequence induced by the sequence of suffixes.

$$\begin{aligned} P(O, Q|\lambda) &= P(O|Q, \lambda)P(Q|\lambda) \\ &= \prod_{t=1}^{T} b_{q_t} \prod_{t=1}^{T} a_{t-1,t} \end{aligned}$$

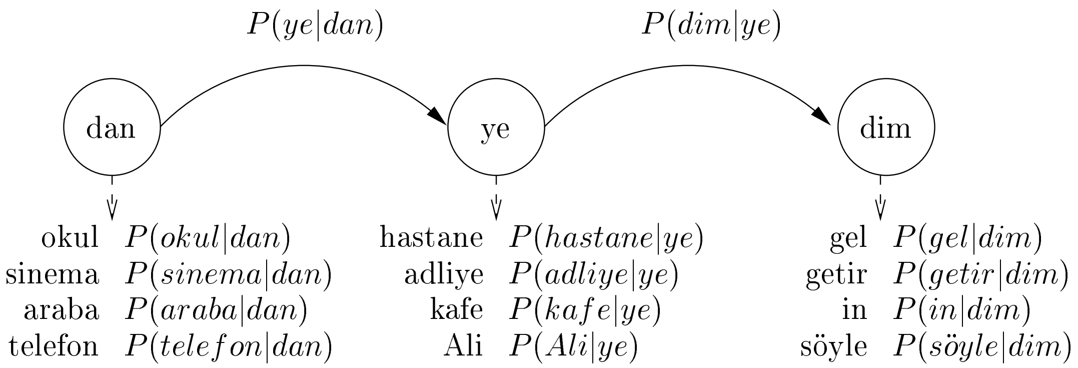


Figure 3.5: A part of a HMM where 1 syllable long suffixes correspond to hidden states and remaining prefix parts correspond to emission symbols.

Figure 3.5 shows a part of a HMM where states are 1-grams of suffixes and emitted symbols are prefixes that remain. Then in Equation 3.12 the state transition probability is $P(S_i|S_{i-2}^{i-1})$ and the observation probability is $P(P_i|S_{i-1}^{i})$. Note that this HMM first predicts the hidden state that corresponds to the next word's suffix and then predicts the next word's prefix using this predicted suffix. This model is built on the assumption that the structure of a sentence is strongly dependent on its suffixes, which is true for Turkish as discussed in the beginning of this chapter.

Model 3: The HMM's assume that the observations are statistically independent. Although the previous model assumes that suffixes are good clues for predicting words, it also assumes that the prefixes which are approximations to roots are independent of each other. Due to the free word order in Turkish this assumption may result in better results, however collocations might be another clue at the word's root level. For instance the root sequence *bakan kurul* which is also the root sequence for *bakanlar kurulu(cabinet)*, is more probable than *bakan bayan(the woman that baby-sits)*.

In this model we modify the previous assumption by assuming independence of suffixes:

$$P(P_i, S_i|P_1^{i-1}, S_1^{i-1}) = P(P_i|P_{i-2}^{i-1})P(S_i|P_{i-1}^{i}) \tag{3.13}$$

Models 4,5,6,7: The previous two models could be trained from a corpus

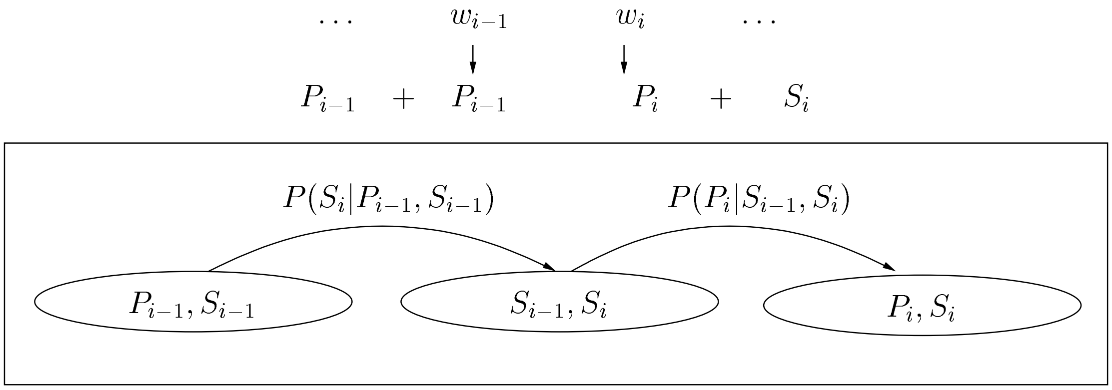


Figure 3.6: Part of a Markov Chain corresponding to the prefix-suffix language model in Equation 3.14

using forward-backward algorithm and can be used for predicting prefixes correponding to a given sequence of suffixes (or vice versa). However in our case both prefix and suffix part of a word are known to us and we can use visible Markov Models, whose states predict the next prefix and suffix alternatively on a given path in the Markov Chain. For instance, the model in Equation 3.14 first predicts the suffix of the next word given the prefix and suffix of the previous word and then predicts the prefix of the next word given the suffix of the next word and suffix of the previous word as in Figure 3.

$$P(P_1^N, S_1^N) = \prod_{i=1}^{N} P(S_i|P_{i-1}, S_{i-1})P(P_i|S_i, S_{i-1}) \quad (3.14)$$

$$= \prod_{i=1}^{N} P(S_i|P_{i-1}, S_{i-1})P(P_i|S_i, P_{i-1}) \quad (3.15)$$

$$= \prod_{i=1}^{N} P(P_i|P_{i-1}, S_{i-1})P(S_i|P_i, S_{i-1}) \quad (3.16)$$

$$= \prod_{i=1}^{N} P(P_i|P_{i-1}, S_{i-1})P(S_i|P_i, P_{i-1}) \quad (3.17)$$

There are 4 possible variants of this approach that are derived by choosing prefix or suffix for prediction given the prefix and suffix of the previous word and choosing the prefix or suffix of the previous word for predicting the remaining part of the next word.

Note that these four language models use the immediate predecessor of the predicted word as an infomation source(1^{st} order Markov), unlike the previous language models which have used previous 2 words(2^{nd} order Markov). 2-gram language models are used in systems where it's too costly to use higher order language models, such as in acoustic component of speech recognition.

Among these four models, we expect two of them to behave different, 3.14 and 3.17. This is because 3.14 favors suffixes whereas the 3.17 favors prefixes as the information source for predicting the next word.

Models 8,9,10,11,12,13,14: The posterior probability of a suffix in Equation 3.11 can be calculated by applying the class-based approach in Equation 2.8 to suffix sequences:

$$P(S_i|S_1^{i-1}) = P(S_i|c(S_i)) \times P(c(S_i)|c(S_1^{i-1})) \quad (3.18)$$

where function $c()$ returns the class of the suffix if its argument is a single suffix, and if its argument is a suffix sequence, returns the sequence of classes that corresponds to it. The probability of a given sequence of prefixes and suffixes will be as in Equation 3.18. Note that we derived this model by replacing the suffixes with their classes and by adding a Markov Property to the emission of suffixes from the suffix classes.

The suffix classes, $c(S_i)$, are computed using the mutual information based clustering algorithm that is introduced in Section [2]. We expect such a clustering algorithm will generate classes that are populated by inflectional suffixes that represent the same linguistic phenomenon and their allomorphs that are generated through various morphographemic rules.

By applying the same class-based approach to our 3-gram Models 1 to 7 we derive the following class models:

$$P(P_1^N, S_1^N) = \prod_{i=1}^{N} P(c(S_i)|c(S_{i-2}^{i-1}))P(P_i|P_{i-2}^{i-1}))P(S_i|c(S_i)) \quad (3.19)$$

$$= \prod_{i=1}^{N} P(c(S_i)|c(S_{i-2}^{i-1}))P(P_i|c(S_{i-1}^{i}))P(S_i|c(S_i)) \quad (3.20)$$

$$
\begin{aligned}
&= \prod_{i=1}^{N} P(P_i|P_{i-2}^{i-1})P(c(S_i)|P_{i-1}^{i})P(S_i|c(S_i)) \\
&= \prod_{i=1}^{N} P(c(S_i)|P_{i-1}, c(S_{i-1}))P(P_i|c(S_i, S_{i-1}))P(S_i|c(S_i)) \\
&= \prod_{i=1}^{N} P(c(S_i)|P_{i-1}, c(S_{i-1}))P(P_i|c(S_i), P_{i-1})P(S_i|c(S_i)) \\
&= \prod_{i=1}^{N} P(P_i|P_{i-1}, c(S_{i-1}))P(S_i|c(S_{i-1}), P_i)P(S_i|c(S_i)) \\
&= \prod_{i=1}^{N} P(P_i|P_{i-1}, c(S_{i-1}))P(S_i|c(P_{i-1}, P_{i-1}))P(S_i|c(S_i))
\end{aligned}
$$

Once we have defined our models we continue with the domain that we will apply our language models.

3.5 Lattice Rescoring for JANUS

JANUS[38] is a speech-to-speech translation system and the speech recognition engine underlying this system is called JRTk[8], stands for JANUS Recognition Toolkit. JRTk is built as a programmable toolkit in order to allow rapid development of speech recognizers for experimenting with different approaches and ideas. JRTk uses a multiple pass decoding schema. Recall that in the decoding phase of the speech recognition task, this kind of recognizers do not use the single best candidate produced by the Viterbi search but rather multiple hypotheses as an *N-best list* or *N-best word lattice* for further refinement. By using such a *multiple-pass* strategy the JRTk speech recognizer employs simple and efficient knowledge sources in the first pass to prune the large search space so that expensive and sophisticated knowledge sources can be used in the second pass to find the globally optimum utterance as in Figure 3.5[17].

The word lattice generated by the Viterbi pass of the decoding is a directed acyclic graph that contains all hypothesized sentences. Each node of the lattice corresponds to a word w_L that starts at frame $F_s(w_L)$. In addition to the start frame and the word itself, the accumulated score of the word in the backtrace of

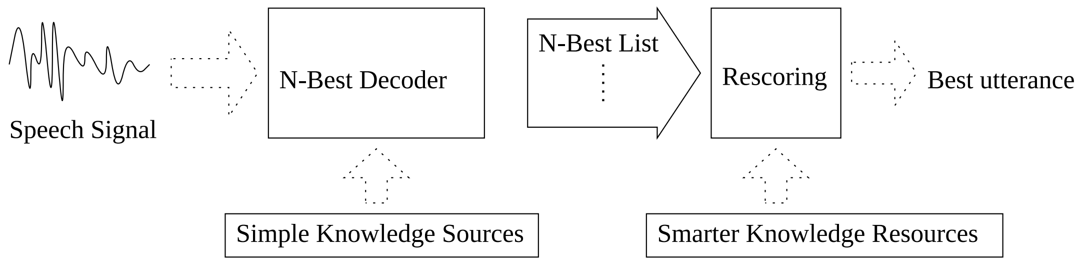


Figure 3.7: Multiple-pass decoding prunes the search space in lower levels to use more sophisticated knowledge resources in higher levels of search

```
%TURN: TU002_66
BEGIN_LATTICE
45 {</s>} -score  2.9E+02
40 {im} -score  6.7E+02 -links { {45 {</s>} 44  3.2E+02}}
35 {lime} -score  4.9E+02 -links { {40 {im} 39 4.8E+02} {45 {</s>} 44  1.5E+02}}
30 {kelime} -score  7.1E+02 -links { {40 {im} 39 4.8E+02} {45 {</s>} 44  2.8E+02}}
20 {dakik} -score  7.1E+02 -links { {40 {im} 39 2.8E+02}{35 {lime} 34  2.8E+02}{45 {</s>} 44  2.8E+02}}
5 {buradaki} -score  7.1E+02 -links { {30 {kelime} 29  3.0E+02}}
5 {bardak} -score  7.5E+02 -links { {20 {dakik} 19 3.3E+02} {35 {lime} 34 5.3E+02}}
5 {bura} -score  7.5E+02 -links { {20 {dakik} 19  4.3E+02} {35 {lime} 34 5.6E+02}{30 {kelime} 29  8.5E+02}}
0 {<s>} -score  1.4E+04 -links { {5 {bura} 4  3.3E+02} {5 {bardak} 4  3.3E+02}{5 {buradaki} 4  3.3E+02}}
END_LATTICE
```

Figure 3.8: A small hypothetical N-best lattice file output of JRTk

the Viterbi pass is also stored on each node. Edges in the word lattice connect a node containing word hypothesis w_L starting at frame $F_e(w_L)$ to all possible successor words in the lattice. The edges in the word lattices have four attributes. Since the recognizer assigns a range of exit frames for each utterance, an edge between word w_L and word w_R stores a possible word exit frame $F_x(w_L)$ for the word w_L and an acoustic score for w_L between the frames $F_e(w_L)$ and $F_x(w_L)$. Also two attributes that uniquely identify the next node, w_R and $F_e(wR)$ are stored on this edge.

In this thesis we are going to use a Turkish speech recognizer[3] that was built using JRTk. Figure 3.10 is a visualized output of this recognizer for utterance *"OSTİMde açılan sergide OSTİMde"*. Figure 3.8 is a hypothetical word lattice output that demonstrates the format of Turkish recognizer's N-best lattices and Figure 3.9 is the corresponding visualization of the lattice. Note that <s> stands for *sentence begin hypothesis* and </s> stands for *sentence end hypothesis*.

The word lattice generated by the recognizer looks like a 1^{st} order Markov Model introduced in the previous chapter, however the probabilities on the edges

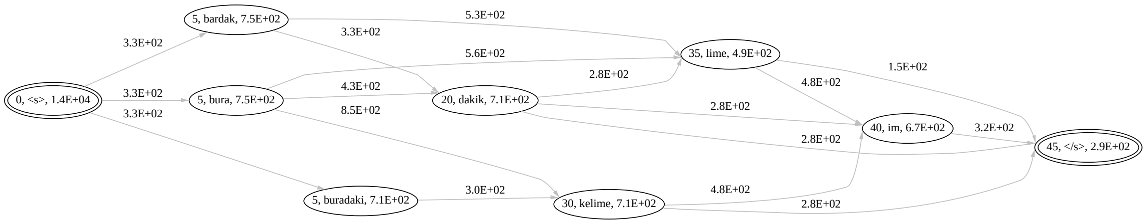


Figure 3.9: A hypothetical word lattice with 16 utterances.

do not sum to 1. It is possible to readily apply a 2-gram language model to compute the edge costs for searching the globally optimal utterance using the Viterbi search or any other shortest path algorithm. In order to use a 3-gram language model we need to increase the context length stored in the nodes to 2 words as described in the previous chapter.

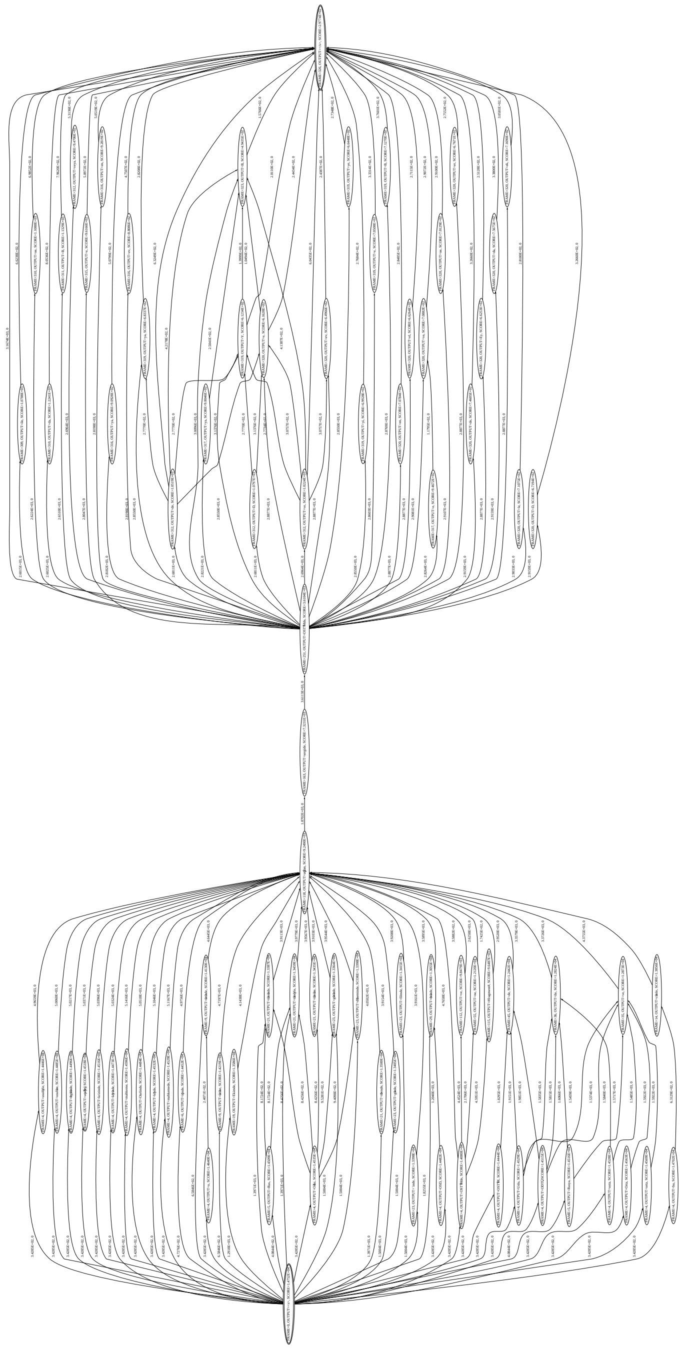


Figure 3.10: A real word-lattice generated by JRtk for the utterance "OSTİMde açılan sergide OSTİMde". The three nodes on the bottleneck of the graph correspond to the last three words of the utterance.

Chapter 4

Experiments and Results

Insanity: doing the same thing over and over again and expecting different results.

Albert Einstein

In this chapter we first introduce the lattice corpus we have used for testing our models. Then we continue the discussion on the language models that were introduced in the previous chapter, elaborating on the training text corpora and experiemental details.

4.1 Lattice Corpus

We have conducted several tests of our language model on the task of selecting the best sequence of words from word lattices generated by JRTk large vocabulary continuous speech recognizer modified for Turkish[3]. The speech recognizer has a reported 16.9% word error rate.

The Turkish recognizer uses a multilevel recognition schema to adjust its vocabulary for each utterance in order to avoid the high out of vocabulary rate[9]. In the first pass the vocabulary of the recognizer is filled with possible roots and in the second pass a new vocabulary of words that are generated from the words recognized in the first pass are used as the vocabulary of the recognizer to run on the same input.

The speech corpus used in the generation of these lattices is a part of GlobalPhone project[29]. The speech samples consist mainly of national and international economics and politics news read by native Turkish speakers. In our corpus we have a total of 123 lattices corresponding to diffent utterances.

4.2 Language Models

4.2.1 Baseline Word-Based Language Model

We have used word 3-gram language models as a baseline system to compare the performance of our models. 3-gram language models approximate the probability of a word sequence as:

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

Although this language model is very commonly used in speech recognition of English, the rapid vocabulary growth of Turkish lowers its performance in our tests. In word based language modeling we used a vocabulary of 64 thousand words, to train our models, and we have included the unknown word token <unk> in this model.

We have trained our language models using a 10M word text corpus that is extracted from an online Turkish newspaper[20, 10, 37]. The articles in the corpus are from various genres including sports, domestic affairs and foreign affairs.

The 3-gram models we have used in our tests are generated by SRI language modeling toolkit[36], which also applies Good-Turing smoothing and Katz Back-off.

4.2.2 Prefix-Suffix Based Language Models

The language models that we have tested are a subset of the models that have been listed in the previous chapter.

The prefix-suffix models discussed in the previous chapter have two variables p and q that denote the length of the prefix and length of the suffix to be extracted from the words. These two variables should be set in a way that the first p units of the words should capture the words' stems and the last q units of the words should capture their most significant suffixes, that is the most significant IGs. In some sample text of 1 million morphologically disambiguated words, Hakkani-Tür has found out that the average root length is 4.10 letters and the average suffix length is 1.71 letters[10], and has got the best results using 4 letter prefix and 3 letter suffix.

A different approach to the issue of prefix and suffix units is to use the first p syllables for prefix and last q syllables for suffix of a word. Since the aim is to set a window size for the significant part of the words, such a variable size window might perform better. The transcription of Turkish is very similar to the phonetic representation and splitting words into syllables is a cheap computational task as described by Solak and Oflazer[32]. In our prefix suffix language models we have adapted this approach, since the syllables correspond to morphemes better than fixed sized suffixes.

The optimal values for prefix and suffix lengths for getting best performance from the language models are likely to be inherent in the language, for this reason we have set the prefix and suffix lengths in accordance with these findings. Since we have used syllable based language models we have set the prefix size to 2 syllables and suffix size to 1 syllable, which roughly correspond to the letter based findings of Hakkani-Tür.

In our prefix and suffix language models we used a prefix vocabulary of size 14000 and suffix vocabulary of size 1000 from the most frequent prefixes and suffixes. We treated the rest of the prefixes and suffixes as unknown tokens

<unk>, and assigned a uniform probability for them.

4.3 Testing

We have implemented an evaluation system for training and testing our models. Figure 4.1 shows the framework of this sytem. The rectangles show data that flows on the arcs, and bold rectangles are inputs of the system. Diamonds represent computations and triangles represent databases where generated data is stored for further processing.

The upper part of this figure represents the training modules of the whole system. In the training modules we have used PERL and C++ for our implementations. The *ngram-class* and *ngram-count* are parts of the SRI language modeling toolkit. *ngram-class* is an implementation of mutual information based clustering and ngram-count is a language model extraction tool that gets n-gram frequencies that are extracted from corpora as its input and construct ngram-models with Good-Turing smoothing and Backoff discounting. The frequency generator produces ngram frequencies for the plain models by doing counting and replaces the suffixes with their class names beforehand if a class based language model is going to be generated.

The lower part of the figure depicts the evaluation component of the whole system. This part has also been implemented using PERL and C++ together. A lattice that is selected for decoding is first converted to Latin character encoding, and the upper case distinction in its hypotheses are removed. The decoder that we have implemented in C++ takes language model pairs and lattices as its input and produces the N best utterances that is induced by them. The language models are applied to the words in the hypothesized words by first incrementing the context of the input lattice from 2 to 3. Then while decoding the lattice using Viterbi algorithm, the prefix and suffix parts of the hypotheses are extracted by using the same routines in the training part for applying the prefix-suffix language models. Note that not all the prefix suffix language models use 3-gram word context, in these cases the lattice context is not incremented.

In order to find the weights of the language model score and the acoustic model score, we have gathered performance data of decoding in a training corpus of 6 lattices for a range of language model weights by generataing N-best lists.

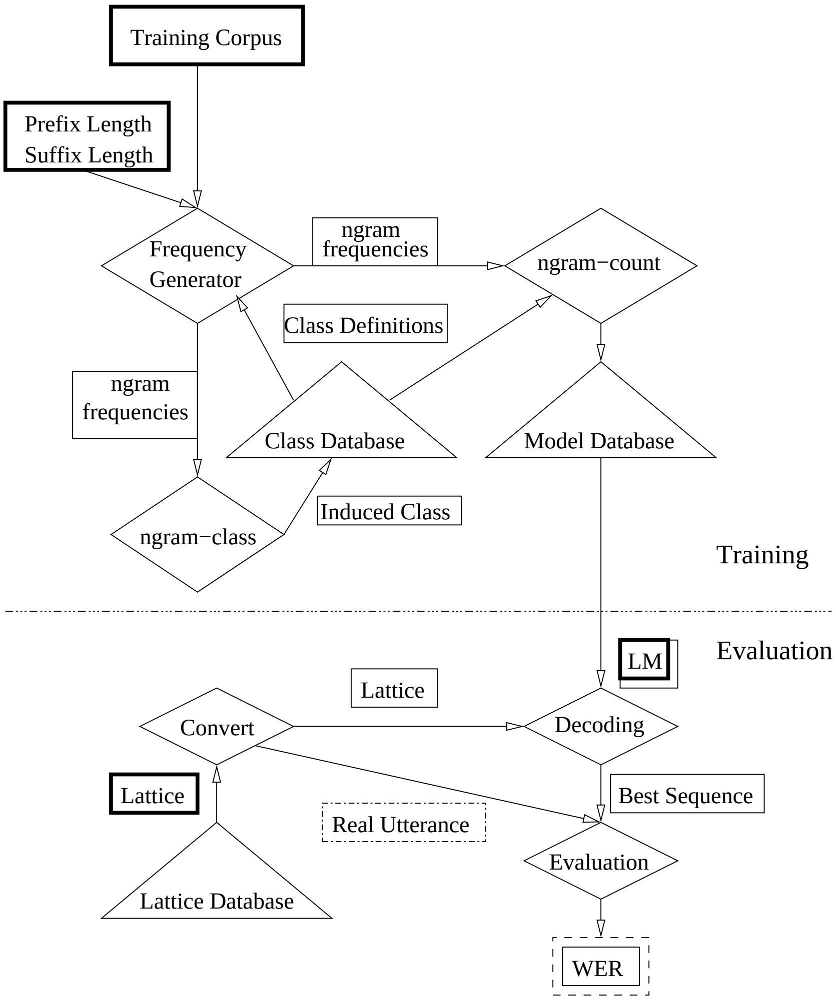


Figure 4.1: The implementation framework of the proposed language models

4.4 Results

In the kingdom of the blind, the one-eyed man is king.

Desiderius Erasmus, Adages

There are acoustic model scores on each lattice that represents the probability of a particular word to be uttered between two frames. The beginning frame of an uttered word is encoded in the state that has the same label and a possible end frame along with the acoustic score of the word between these two frames is encoded in the links of this state. We have weighted these acoustic model scores and our language model scores in order to compute the best hypothesis in the lattice.

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(A|W)^{\alpha} \times P(W)^{\beta} \tag{4.1}$$

We have chosen acoustic model weight α and language model weight β, such that $\alpha + \beta = 1$. We have determined optimal α and β values to be 0.02 and 0.98 for our 2-gram models and 0.10 and 0.90 for our 3-gram models, through training our models on a 6 lattice training set.

Acoustic WER	54.51%
Recognition Recall	90%

Table 4.1: Performance of the recognizer using only acoustic model scores.

4.4.1 Initial Performance of the Recognizer

The performance of the recognizer when only acoustic model scores are used is shown in Table 4.1. Among the 20 lattices that we used in our test set 2 lattices did not contain the corresponding correct utterance, and the average WER using the acoustic model was 54.51%.

4.4.2 Baseline Word-based Language Model Performance

We have used 64M vocabulary 3-gram and 2-gram language models to compare our results. The performance of 3-gram and 2-gram language models are shown in Table 4.2. We have trained our word based language model on a 10 million word corpus, using a 64 thousand word vocabulary of most common words in this corpus.

	WER
Word 2-gram	39.78 %
Word 3-gram	31.65 %

Table 4.2: Performance of the recognizer with word based 2-gram and 3-gram language models.

4.4.3 Performance of the Prefix Suffix based Language Models

We have tested 7 prefix suffix models in our experiments.Table 4.3 lists these language models. The performance of these models is shown in Table 4.5.

Model 1	$P(P_i \mid P_{i-2}, P_{i-1}) \times P(S_i \mid S_{i-2}, S_{i-1})$
Model 2	$P(S_i \mid S_{i-2}, S_{i-1}) \times P(P_i \mid S_{i-1}, S_i)$
Model 3	$P(P_i \mid P_{i-2}, P_{i-1}) \times P(S_i \mid P_{i-1}, P_i)$
Model 4	$P(S_i \mid P_{i-1}, S_{i-1}) \times P(P_i \mid S_i, S_{i-1})$
Model 5	$P(S_i \mid P_{i-1}, S_{i-1}) \times P(P_i \mid S_i, P_{i-1})$
Model 6	$P(P_i \mid P_{i-1}, S_{i-1}) \times P(S_i \mid P_i, S_{i-1})$
Model 7	$P(P_i \mid P_{i-1}, S_{i-1}) \times P(S_i \mid P_i, P_{i-1})$

Table 4.3: The models that we have tried in our experiments

Since our aim was to compare the size of the models as well as their performance with the baseline model, we have given emphasis on this issue in our experiments. We have used a small vocabulary size for our prefix suffix language models and a large vocabulary size for word based baseline language models to demonstrate the robustness of our models. The n-gram frequencies stored in the models vary directly proportional to the size of the vocabulary that generates them. The size of n-grams decrease as we go from words to suffixes. The 3-gram counts of prefixes and words decrease due to data sparseness. There are too many 3-grams of prefixes and words but only a small proportion of them have significant probability mass that will be beneficial for the language models. For this reason the total number of 3-grams in these models are less than the total number of their 2-grams. The size values for models is given in Table 4.4.

	1-gram	2-gram	3-gram
Word-based	64,001	2,177,344	1,074,901
Model 1	15,001	1,601,291 + 19,766	672,758 + 67,450
Model 2	15,001	19,766 + 112,917	67,450 + 246,232
Model 3	15,001	1,601,291 + 621,323	672,758 + 658,186
Model 4	15,001	85,063 + 112,917	260,156 + 246,232
Model 5	15,001	85,063 + 215,539	260,156 + 658,186
Model 6	15,001	310,851 + 151,790	636,913 + 246,523
Model 7	15,001	310,851 + 621,323	636,913 + 658,186

Table 4.4: Size of language models used in experiments in terms of n-gram frequencies they contain.

	Context	WER
Model 1	3-gram	40.41%
Model 2	3-gram	37.18%
Model 3	3-gram	40.41%
Model 4	2-gram	44.20%
Model 5	2-gram	41.04%
Model 6	2-gram	44.24%
Model 7	2-gram	48.57%

Table 4.5: Performance of our prefix suffix models.

4.5 Discussions

The previous subsections contain interesting results about the performance of our recognizer. First of all, the 2-gram Model 5 performs as well as the 3-gram Model 1. This is probably because of the assumption in Model 5 that suffixes and prefixes are generated from independent random processes. Unlike Model 1, Model 5 respects a statistical correlation between prefix and suffix parts of words.

The performance of Model 7 is interestingly worse than the other 2-gram models. This model is different from the other 2-gram models in its assumption that the suffix of the next word is dependent on the prefix of the previous word and prefix of the next word. This is due to the productive suffixation in Turkish. Let us consider the following prefix sequence:

araş dosya

One can routinely generate many legal and highly probable Turkish phrases by appending suffixes to the prefixes in the above sequence. Thus Model 7 can only assign a nearly uniform probability to all these possible suffixations, which will not differenctiate between the hypotheses.

Among the 3-gram language models Model 2 performs best. Recall that this model was a HMM over the hidden suffix states that generate prefixes. The high performance of this model is evidence of the fact that in Turkish suffixes contain a larger amount of information than do the prefixes in 3 word phrases.

Also in our results we have found a 2-gram prefix suffix language model(Model 5), that has similar performance to a word 2-gram language model. The lexical tree search at early stages of the speech recognition that has high memory demand, requires effective search space pruning techniques to cope with this problem. Language model lookahead [6] is one technique that is used in most state of the art speech recognizers for discriminating between competing hypothesis with similar acoustic scores at this stage of recognition. The identity of a hypothesis is known as the last phone of the word is recognized by the reconizer, and the first phone of the probable words that the language model determines is instantiated for recognizing the next word. Since new words are instantiated at the beginning of every frame in the acoustic signal, even with the language model lookahead there is a huge memory requirement.

If the word-based 2-gram language model is used, it will consume a lot of memory at a time it is most needed. Model 5, whose size is 55% of the size of the word 2-gram model, can be used in language model lookahead instead of a word 2-gram language model in this stage.

Chapter 5

Conclusion

In this thesis we have presented a set of tools for language modeling of Turkish through statistical methods.

We have tried several statistical language models that use subword units to model language. Using subword units for modeling reduces the probability of assigning scores to previously unseen words in the corpus. That is subword language modeling units decrease the out of vocabulary rate. But this does not translate into a decrease in the WER, because of the more uniform distribution of the subword units when compared to distribution of words themselves.

The performance of our subword language models demonstrate a stronger correlation between the surface word sequence and the suffixes of these words than with the prefixes.

Although we could not achieve a remarkable decrease in the word error rate through using subword units in language modeling, we have reached the performance of a large word based language model with a half sized subword model in 2-gram word context. This language model can be used in acoustic decoding part of the speech recognizer for pruning the large search space for a Turkish speech recognizer.

Bibliography

[1] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions in Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[2] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-Based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[3] K. Çarkı, P. Geutner, and T. Schultz. Turkish LVCSR: Towards better speech recognition for agglutinative languages. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, Phoenix, Mar. 2000.

[4] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for Part-of-Speech Tagging. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 784–789, Washington, D.C., July 1993. AAAI Press.

[5] K. W. Church and W. A. Gale. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54, 1991.

[6] N. Deshmukh, G. Aravind, and J. Picone. Hierarchical search for large-vocabulary conversational speech recognition. *IEEE Transactions on Signal Processing*, 16, Sept. 1999.

[7] E. E. Erguvalı. *The Function of Word Order in Turkish Grammar*. PhD thesis, University of California, Los Angeles, 1979.

[8] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal. The karlsruhe-verbmobil speech recognition engine. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 83–86, Munich, Apr. 1997.

[9] P. Geutner, M. Finke, and A. Waibel. Selection criteria for hypothesis driven lexical adaptation. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 617–620, Phoenix, AZ, Mar. 1999.

[10] D. Z. Hakkani-Tür. *Statistical Language Modeling for Turkish*. PhD thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2000.

[11] D. Z. Hakkani-Tür and K. Oflazer. Statistical Morphological Disambiguation for Agglutinative Languages. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, Aug. 2000.

[12] J. Hankamer. Lexical Representation and Process. In W. Marslen-Wilson, editor, *Morphological Parsing and the Lexicon*. The MIT Press, 1989.

[13] P. Heeman and J. Allen. Incorporating POS tagging into language modeling. In G. Kokkinakis, N. Fakotakis, and E. Dermatas, editors, *Proceedings of the 5th European Conference on Speech Communication and Technology*, Rhodes, Greece, Sept. 1997.

[14] P. Heeman and J. Allen. Intonational Boundaries, Speech Repairs, and Discourse Markers: Modeling Spoken Dialog. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, July 1997.

[15] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1998.

[16] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, 1980.

[17] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.

[18] S. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, March 1997.

[19] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[20] Milliyet. Turkish daily newspaper. http://www.milliyet.com.tr.

[21] K. Oflazer. Two-level Description of Turkish Morphology. *Literary and Linguistic Computing*, 8(3), 1993.

[22] K. Oflazer. Dependency Parsing with an Extended Finite State Approach. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, University of Maryland, College Park, MD, June 1999.

[23] K. Oflazer, D. Z. Hakkani-Tür, and G. Tür. Design for a Turkish Treebank. In *Proceedings of Workshop on Linguistically Interpreted Corpora, EACL'99*, Bergen, 1999.

[24] J. R. Pierce. *An introduction to information theory: symbols, signals & noise*. Dover Publications, 1980.

[25] L. R. Rabiner and B. H. Juang. An Introduction to Hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, Jan. 1986.

[26] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? In *Proceedings of the IEEE*, volume 88(8), 2000.

[27] S. J. Russell and P. Norvig. *Artificial Intelligence. A Modern Approach.* Prentice-Hall, Englewood Cliffs, 1995.

[28] T. Schultz and A. Waibel. Language portability in acoustic modeling. In *Proceedings of the Workshop on Multilingual Speech Communication*, Kyoto, 2000.

[29] T. Schultz, M. Westphal, and A. Waibel. The GlobalPhone Project: Multilingual LVCSR with Janus-3. In *Proceedings of SQEL*, pages 20–27, Plzen, 1997.

[30] R. Schwartz and Y.-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the n most likely sentence hypotheses. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 81–84, Albuquerque, NM, Apr. 1990.

[31] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[32] A. Solak and K. Oflazer. A finite state machine for Turkish syllable structure analysis. In *Proceedings of the Fifth International Symposium on Computer and Information Sciences*, volume 2, pages 1195–1202, Cappadocia, 1990.

[33] A. Solak and K. Oflazer. Design and implementation of a spelling checker for turkish. *Literary and Linguistic Computing*, 1993.

[34] R. R. Stanley Chen, Douglas Beeferman. Evaluation metrics for language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, 1998.

[35] A. Stolcke. Linguistic knowledge and empirical methods in speech recognition. *AI Magazine*, 1997.

[36] A. Stolcke. SRILM—the SRI language modeling toolkit. http://www.speech.sri.com/projects/srilm/, 1999.

[37] G. Tür. *A Statistical Information Extraction System for Turkish*. PhD thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2000.

[38] A. Waibel and et. al. JANUS - a speech-to-speech translation system using connectionist and symbolic processing strategies. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 793–796, Toronto, May 1991.

[39] S. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, 1996.