INTEGRATING MORPHOLOGY INTO AUTOMATIC SPEECH RECOGNITION: MORPHOLEXICAL AND DISCRIMINATIVE LANGUAGE MODELS FOR TURKISH

by
Haşim Sak
B.S., Computer Engineering, Bilkent University, 2000
M.S., Computer Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in
Boğaziçi University
2012

INTEGRATING MORPHOLOGY INTO AUTOMATIC SPEECH RECOGNITION: MORPHOLEXICAL AND DISCRIMINATIVE LANGUAGE MODELS FOR TURKISH

APPROVED BY:

Assoc. Prof. Tunga Güngör ....
(Thesis Supervisor)

Assoc. Prof. Murat Saraçlar ....
(Thesis Co-supervisor)

Assist. Prof. Deniz Yüret ....

Prof. Fikret Gürgen ....

Prof. Lale Akarun ....

DATE OF APPROVAL: 11.11.2011

ACKNOWLEDGEMENTS

I am more than grateful to my excellent supervisors, Tunga Güngör and Murat Saraçlar for their great contributions to this thesis, invaluable guidance in this path and approaching me always with support, encouragement and understanding.

I would like to thank members of my thesis committee, Lale Akarun, Fikret Gürgen and Deniz Yüret for their invaluable feedbacks and contributions to this work. I would like to also thank Kemal Oflazer, Deniz Yüret, and Dilek Hakkani-Tür for providing me with data and tools.

I would like to thank especially Ebru Arısoy for her scientific contributions to this thesis by providing resources, tools and experimental set up facilitating this work greatly. I would like to also thank Sıddıka Parlak, İpek Şen, Erinç Dikici, and Doğan Can at BÜSİM lab for being my friends and for their help and contributions to this thesis.

My special thanks go to my friends at CMPE. Ahmet Yıldırım, Akın Günay, Alp Kındıroğlu, Arda Çelebi, Barış Gökçe, Barış Kurt, Barış Evrim, Başak Aydemir, Can Kavaklıoğlu, Çetin Meriçli, Dağhan Dinç, Ergin Özkucur, Furkan Kıraç, Gaye Genç, Itır Karaç, İlker Yıldırım, İsmail Arı, Nadin Kökçiyan, Nuri Taşdemir, Onur Güngör, Özgür Kafalı, Roza Ghamari, Seniha Köksal, Serhan Danış, Suzan Bayhan, Tekin Meriçli, Yunus Emre Kara and many others have made Boğaziçi such a fun and friendly place. I would like to also thank the faculty members of CMPE, especially Suzan Üsküdarlı, Lale Akarun, Cem Ersoy and Tuna Tuğcu for being the source of exceptional positive energy at CMPE.

I would like to express my gratitude to my parents and brothers, especially my twin brother Halis for his support, understanding and help throughout my life.

I would like to thank my dear friends Nagehan Aktunç, Ersin Tuşgul and Ahmet Bulut, and I feel very lucky for having them in my life.

My last, and most heartfelt acknowledgment must go to Derya Çavdar for her love and support. I feel blessed with her presence in my life.

This thesis was supported in part by the Scientific and Technical Research Council of Turkey (TÜBİTAK) under BİDEB 2211, and grant numbers 107E261,105E102 and 109E142, in part by Boğaziçi University Research Fund (BAP) under grant numbers 06A102, 08M103, and in part by Turkish State Planning Organization (DPT) under the TAM project number 2007K120610.

ABSTRACT

INTEGRATING MORPHOLOGY INTO AUTOMATIC SPEECH RECOGNITION: MORPHOLEXICAL AND DISCRIMINATIVE LANGUAGE MODELS FOR TURKISH

Languages with agglutinative or inflectional morphology prove to be challenging for speech and language processing due to relatively large vocabulary size leading to a high number of out-of-vocabulary (OOV) words. In this thesis, we tackle with these challenges in automatic speech recognition (ASR) frame for Turkish having an extremely productive inflectional and derivational morphology. First, we build the necessary tools and resources for Turkish, namely a finite-state morphological parser, a perceptron-based morphological disambiguator, and a text corpus collected from web. Second, we introduce two complementary language modeling approaches to alleviate OOV word problem and to exploit morphology as a knowledge source. The first model, *morpholexical language model*, is a generative n-gram model, where modeling units are lexical-grammatical morphemes instead of commonly used words or statistical sub-words. We also propose a novel approach for integrating the morphology into an ASR system in the finite-state transducer framework as a knowledge source. The second model is a linear reranking model trained discriminatively with a variant of the perceptron algorithm, word error rate (WER) sensitive perceptron, using morpholexical and morphosyntactic features to rerank n-best candidates obtained with the generative model. We apply the proposed models in Turkish broadcast news transcription task and give experimental results. The morpholexical model is highly effective in alleviating OOV problem and improves the WER over word and statistical sub-word models by 1.8% and 0.8% absolute respectively. The discriminatively trained model further improves the WER of the system by 0.8% absolute. Finally, we present an algorithm for on-the-fly lattice rescoring with low-latency.

ÖZET

BİÇİMBİLİMİN OTOMATİK KONUŞMA TANIMAYA BÜTÜNLEŞTİRİLMESİ: TÜRKÇE İÇİN BİÇİMSÖZLÜKSEL VE AYIRICI DİL MODELLERİ

Göreceli olarak geniş bir dağarcığa sahip sondan eklemeli ya da çekimsel biçimbilime sahip diller konuşma ve dil işlemede yüksek sayıda dağarcık dışı (DD) kelimenin görülmesine neden olduğundan bazı zorluklar sunar. Bu tezde, bu zorluklar ile otomatik konuşma tanıma (OKT) kapsamında çok üretken çekimli ve türevsel biçimbilime sahip olan Türkçe için ilgileniyoruz. İlk olarak, Türkçe için gereken kaynakları ve araçları oluşturduk. Bunlar sonlu-durum biçimbilimsel çözümleyici, perceptron-tabanlı biçimbilimsel tekleştirici, ve metin derlemidir. İkinci olarak, DD kelime sorununu gidermek ve biçimbilimsel bilgiden kaynak olarak yararlanmak için birbirini tamamlayan iki dil modeli yaklaşımı geliştirdik. İlk model, sıklıkla kullanılan kelime ve kelime-altı birimler yerine sözlüksel-dilbilgisel biçimbirimleri kullanan üretici n-birimli bir model olan *biçim-sözlüksel dil modelidir.* Ayrıca, sonlu durum dönüştürücü çerçevesinde biçimbilimi bilgi kaynağı olarak OKT sistemine entegre etmek için yeni bir yöntem sunduk. İkinci model, üretici model ile elde edilen en iyi adayları tekrar sıralamak için biçim-sözlüksel ve biçim-dizimsel öznitelikleri kullanan kelime hata oranı (KHO) duyarlı algılayıcı bir algoritma ile ayırıcı olarak eğitilmiş doğrusal bir modeldir. Önerilen yöntemleri haber kayıtlarının yazılandırılması görevine uyguladık ve deneysel sonuçlar elde ettik. Biçim-sözlüksel model dağarcık dışı kelime sorununu nispeten gidermiş ve konuşma tanımada kelime hata oranını kelime ve istatistiki kelime-altı modellere göre sırasıyla %1.8 ve %0.8 oranında iyileştirmiştir. Ayırıcı olarak eğitilmiş model sistem başarımını %0.8 oranında daha da iyileştirmiştir. Son olarak, konuşma tanıma çıktısı olan kelime örgülerini tanıma yapılırken tekrar değerleyen bir algoritma geliştirdik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Acoustic feature vector
A	Acoustic observations - acoustic feature vector sequences
$count(\cdot)$	Number of occurrences for an n-gram
$det(\cdot)$	Determinization operation
$\mathbf{GEN}(x)$	A function generating hypotheses for an input x
E	A set of transitions
F	The set of final states
I	The set of initial states
$\mathbb{K}$	A semiring
$min(\cdot)$	Minimization operation
$L(\cdot)$	Log-likelihood function
$P(\cdot)$	Probability function
$P(\cdot\|\cdot)$	Conditional probability function
Q	A set of states
$\mathbb{R}$	The set of real numbers
T	A weighted finite-state transducer
$\mathcal{V}$	Vocabulary
w	A word
W	A sequence of words
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$Z(\cdot)$	Normalization function for a parameter
$\vec{\alpha}$	Parameter vector
Δ	Delta coefficients
Δ	Output alphabet
$\Delta\Delta$	Delta-delta coefficients
$\vec{\gamma}$	Averaged parameter vector
λ	The initial weight function

$\pi_\epsilon(\cdot)$	Auxiliary symbol removal operation
ρ	The final weight function
Σ	Input alphabet
θ	Model parameters
$\otimes$	Product operation to compute the weight of a path
$\oplus$	Sum operation to compute the weight of a sequence
$\cdot \circ \cdot$	Finite-State Transducer Composition operator

LIST OF ACRONYMS/ABBREVIATIONS

ASR	Automatic Speech Recognition
BN	Broadcast News
DLM	Discriminative Language Model
FLM	Factored Language Model
FSA	Finite-State Automaton
FSM	Finite-State Machine
FST	Finite-State Transducer
GCLM	Global Conditional Log-Linear Model
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HTK	Hidden Markov Toolkit
IG	Inflectional Group
LM	Language Model
LVCSR	Large-Vocabulary Continuous Speech Recognition
Max-Ent	Maximum Entropy
MDL	Minimum Description Length
ME	Maximum Entropy
MFCC	Mel Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
OOV	Out-of-Vocabulary
PoS	Part-of-Speech
RTF	Real-Time Factor
SRILM	SRI Language Modeling
WER	Word Error Rate
WFST	Weighted Finite-State Transducer

1. INTRODUCTION

The capabilities and applications of speech and language processing (SLP) have increased substantially in recent years. Automatic speech recognition, speech to speech and machine translation, spoken human-computer interaction, speech indexing and information extraction are a few widely known examples from its wide range of applications. SLP systems have been indispensable in our daily lives through technologies such as personal computers, internet search engines and mobile phones. The research in this area of science and technology has been continuously pushing the state-of-the-art to achieve better accuracy in SLP systems. Spoken human languages possess different characteristics that prove to be challenging for speech and language processing in some languages. One such characteristic is agglutinative or inflective morphology. The complex morphology or word structure has been an important factor reducing the accuracy of SLP systems in morphologically rich languages such as Arabic, Czech, Finnish, Korean and Turkish.

Turkish is an agglutinative language with a productive inflectional and derivational morphology. In agglutinative languages, new words can be formed by stringing morphemes - stems and suffixes - together. Therefore, in such languages, words have some internal structure, representing syntactic and semantic morphological or grammatical features.

In this thesis, our aim is to solve the morphology related problems in speech and language processing of Turkish and further exploit the morphological information to increase the accuracy of SLP systems. We develop a set of tools, resources and methodologies for efficient and effective processing of morphologically rich languages to increase the performance of SLP systems. We primarily focus on Turkish, however the techniques are applicable to other languages with agglutinative or inflective morphology. Although, the application of the proposed methods concentrates on spelling correction and language modeling for speech recognition, they are applicable to other areas such as machine translation, language learning, and language generation.

1.1. Motivation

Statistical language models are one of the commonly used components in speech and language processing systems, which are concerned with morphology most. They are used for instance in speech recognition, machine translation and spelling correction. The word n-gram language models are the most common statistical language models and they are used to assign probabilities to word sequences. In morphologically simpler languages like English, word n-gram language models have been very successful. On the other hand, language modeling for morphologically rich languages such as Arabic, Czech, Finnish, and Turkish proves to be challenging. The out-of-vocabulary (OOV) rate for a fixed vocabulary size is significantly higher in these languages, since there are a large number of words in language vocabulary due to productive morphology. The OOV rate is important for natural language processing applications since, for instance, the higher OOV rate leads to higher word error rate (WER) in ASR systems, and OOV words cannot be translated in machine translation (MT) systems. Having a large number of words also contributes to high perplexity numbers for standard n-gram language models due to data sparseness. These problems are especially pronounced for Turkish as being an agglutinative language with a highly productive inflectional and derivational morphology.

We can reduce the OOV rate by increasing the vocabulary size if it is not limited for instance by the size of the text corpus or parallel corpus available for ASR and MT systems. However, this also increases the computational and memory requirements of the system. Besides, it may not lead to significant performance improvement due to data sparseness problem of insufficient data for robust estimation of language model parameters. Therefore, to overcome the high growth rate of vocabulary and the OOV problem, using grammatical or statistical sub-lexical units for language modeling has been a common approach. The grammatical sub-lexical units can be morphological units such as morphemes or some grouping of them such as stem and ending (grouping of suffixes). The statistical sub-lexical units can be obtained by splitting words using statistical methods.

Morphological processing of languages with complex morphology may prove to be useful to extract and exploit the information hidden in the word structure for SLP applications. This is motivated by the fact that in such languages, grammatical features and functions associated with the syntactic structure of a sentence in morphologically poor languages are often represented in the morphological structure of a word in addition to the syntactic structure. Therefore, morphological parsing of a word may reveal valuable information in its constituent morphemes annotated with morphosyntactic and morphosemantic features to exploit for language modeling.

The motivation for this thesis can be summarized as follows:

- Languages with productive morphology tend to have a large number of words in their vocabularies. Hence, the out-of-vocabulary (OOV) rate for a fixed vocabulary size is significantly higher in these languages. Higher OOV rates cause lower accuracy in SLP systems.
- Turkish has in theory unlimited vocabulary due to iteration of some suffixes like causative suffix.
- Having a large number of words also increases data sparsity problem, which prevents reliable estimation of model parameters with sparse data.
- Word-based n-gram language models are not satisfactory for morphologically rich languages due to OOV and data sparsity problem mentioned above.
- Statistical sub-lexical units can solve OOV problem but the loss of linguistic information with statistical units can harm the language model and it makes it harder to integrate morphological features in the first-pass or second-pass model.
- Using a morphological parser to obtain grammatical morphemes is a better approach. However, surface form morphemes may not be a good choice for some languages having inter-morpheme coarticulation problem, such as Korean. Hence, the optimal method should enable using a pronunciation lexicon with sub-lexical units.
- The lexical morphemes as the linguistic construction units for words are natural and optimal units in language modeling. For instance, a finite-state transducer model for Turkish morphology operates on lexical morphemes.

- Standard n-gram language models over the lexical morphemes can be estimated and these models can be efficiently represented with weighted finite-state transducers.
- Using lexical morphemes enables us to combine computational pronunciation lexicons with n-gram language models over these lexical morphemes.
- If the small size of lexical morphemes hurts the language model robustness, they can be combined to form longer units, such as lexical stem+ending. This effectively solves the n-gram history coverage problem of short sub-lexical units.
- The lexical morphemes can also carry syntactic and semantic morphological features. This information can be exploited with feature-based methods, such discriminative reranking and maximum entropy models.
- The lexical morphemes greatly alleviate the OOV problem since any word that can be morphologically analyzed is now effectively in the vocabulary of the system. This, for instance, provides an unlimited vocabulary for Turkish speech recognition.
- The lexical morphemes require some tools and resources for morphological processing of a language. We need a morphological parser to analyze the words, a morphological disambiguator to choose the correct analysis among ambiguous parses, and a text corpus to estimate the parameters of the language model.
- The language models based on lexical morphemes can be constrained with the lexical transducer of the morphological parser to generate only valid morpheme sequences and hence valid word forms as output from the system. Therefore, the lexical morphemes can prevent the over-generation problem of sub-lexical units efficiently and effectively.

1.2. Approach and Contributions

This thesis presents a morphology oriented linguistic approach for language modeling in morphologically rich languages as an alternative to word and sub-word based models. In this thesis, we first built a set of resources and tools for morphological processing of Turkish. The tools and language resources as given below are available for research purposes[1] :

- A stochastic finite-state morphological parser: It is a weighted lexical transducer that can be used for morphological analysis and generation of words. The transducer has been stochastized using the morphological disambiguator and the web corpus. This parser is used to obtain the linguistic segmentations of words in this thesis.
- An averaged perceptron-based morphological disambiguator: The proposed system has the highest disambiguation accuracy reported in the literature for Turkish. It also provides great flexibility in features that can be incorporated into the disambiguation model, parameter estimation is quite simple, and it runs very efficiently. The disambiguator is used for resolving the ambiguities in morphological analysis of words to obtain a disambiguated corpus for training the statistical models in the thesis.
- A web corpus (a corpus collected from the web): We aimed at collecting a representative sample of the Turkish language as it is used on the web. This corpus is the largest web corpus for Turkish. A part of this corpus is used for building the language models for broadcast news transcription task in this thesis.

Standard n-gram language models are difficult to beat if you have enough data. They also lead to efficient dynamic programming algorithms for decoding due to local statistics, and they can be efficiently represented as deterministic weighted finite-state automata [4]. In this thesis, we propose a novel approach for language modeling of morphologically rich languages. The proposed model as we call it *morpholexical language model* can be considered as a linguistic sub-lexical n-gram model in contrast to statistical sub-word models. The morpholexical n-gram language model is superior to word n-gram models in the following aspects.

- The vocabulary is unlimited since the modeling units are sub-lexical units.
- The OOV rate is effectively reduced to about 1.3 per cent on the test set. For comparison, the 200K word model has about 2 per cent OOV rate.

[1] All resources are available at http://www.cmpe.boun.edu.tr/~hasim

- The perplexity on the test set is lower than word models since it alleviates data sparsity problem.

Besides, it is superior to statistical sub-word n-gram models in some other aspects.

- The modeling units as being lexical and grammatical morphemes provide a linguistic approach.
- The linguistic approach enables integration with other finite-state models like pronunciation lexicon.
- It generates only valid word forms when composed with a computational lexicon.
- The lexical and morphosyntactic features can be further exploited in a rescoring or reranking model.

In this thesis, we propose a novel approach to build a *morphology-integrated search network* for ASR with unlimited vocabulary in the weighted finite-state transducer framework (WFST). The proposed *morpholexical search network* is basically obtained by the composition of the lexical transducer of the morphological parser and the transducer of morpholexical language model. This model has the advantage of the dynamic vocabulary in contrast to word models and it only generates valid word forms in contrast to sub-word models. The proposed model improves ASR word error rate by 1.8 per cent absolute over word models and 0.8 per cent absolute over statistical sub-word models at $\sim$ 1.5 real-time factor.

We further improve ASR performance by using morpholexical and morphosyntactic features in a discriminative n-best hypotheses ranking framework with a variant of the perceptron algorithm. The perceptron algorithm is tailored for reranking recognition hypotheses by introducing error rate dependent loss function. The improvements of the first-pass in WER are mostly preserved in the rescoring as 2.2 per cent absolute over word models and 0.7 per cent absolute over statistical sub-word models.

We also present an on-the-fly lattice rescoring algorithm for low-latency real-time speech recognition. The algorithm enables us to rescore the recognition lattices with a better language model on-the-fly while generating the lattice in the decoder.

1.3. Organization of the Thesis

The presentation of this thesis is organized as follows: In Chapter 2, we give an overview of speech and language processing techniques that we use in this thesis. This Chapter also describes the Turkish broadcast news transcription system on which we carry out our experiments. Besides, the related work on language modeling and speech recognition is given here. In Chapter 3, we introduce the methods, tools and resources that we have built for morphological processing and language modeling of Turkish. In Chapter 4, we present the morpholexical and discriminative language modeling approaches that we propose for language modeling of Turkish. In Chapter 5, we propose an algorithm for rescoring recognition lattices on-the-fly for low-latency real-time speech recognition. Finally, in Chapter 6, we conclude the presentation with a summary and discussion of contributions and findings.

2. BACKGROUND AND SYSTEM DESCRIPTION

In this chapter, we introduce the underlying techniques, methods, systems and algorithms on which the proposed methods in this dissertation rests. We also give an overview of the related work and describe the system on which the experiments are carried out.

2.1. Weighted Finite-State Transducers

Weighted finite-state transducers (WFSTs) are widely used in speech and language processing applications [5–8]. WFSTs are finite-state machines in which each transition is augmented with an output label and some weight, in addition to the familiar (input) label in finite-state automata [9–13]. The weights may represent probabilities, log-likelihoods, or they may be some other costs used to rank alternatives.

The weights are, more generally, elements of a semiring $(\mathbb{K}, \oplus, \otimes, 0, 1)$, that is a ring that may lack negation [13]. The $\otimes$-operation is used to compute the weight of a path by $\otimes$-multiplying the weights of the transitions along that path. The $\oplus$-operation computes the weight of a pair of input and output strings (x, y) by $\oplus$-summing the weights of the paths labeled with (x, y). Some familiar semirings are the tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ related to classical shortest-paths algorithms, and the probability semiring $(\mathbb{R}, +, \times, 0, 1)$. The following gives a formal definition of weighted transducers as given in [14].

Definition 2.1. *A weighted finite-state transducer T over a semiring $(\mathbb{K}, \oplus, \otimes, 0, 1)$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where: Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I \subseteq Q$ the set of initial states; $F \subseteq Q$ the set of final states; $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ a finite set of transitions; $\lambda : I \to \mathbb{K}$ the initial weight function; and $\rho : F \to \mathbb{K}$; the final weight function mapping F to $\mathbb{K}$. Weighted finite-state automata can be defined in a similar way by simply omitting the output labels or making the input and output labels the same. Similarly, finite-state transducers (FSTs) can be defined by omitting* *the weights.*

The weighted finite-state transducers provide a common and natural representation and algorithmic framework for speech recognition [5, 6, 8, 15]. The major components of speech recognition systems, including hidden Markov models (HMMs), context-dependency models, pronunciation lexicons, statistical grammars or language models, and the speech recognition output of word or phone lattices can all be represented as weighted finite-state transducers or automata. This framework also provides general algorithms and operations for building, combining and optimizing these transducer models, including composition for model combination, weighted determinization and minimization for time and space optimization of models, and a weight pushing algorithm for redistributing transition weights optimally for speech recognition.

A stochastic grammar or n-gram language model commonly used in speech recognition G can be represented compactly by a finite-state transducer [4]. For example, Figure 2.1 illustrates a finite-state representation strategy for a back-off bigram language model. This model has a state w_i for each unigram word history. A transition from a state w_i to w_j corresponds to a bigram w_iw_j seen in the training corpus, and it has the label $w_j : w_j$ and the weight $P(w_j|w_i)$ for the estimated bigram probability. The bigrams not seen in the training data are represented by backing-off to a state b having no word history. This models back-off strategy to a lower order language model, in this case, a unigram model, which is used for smoothing. The transition from a state w_i to a back-off state b has the transition probability $\beta(w_i)$ which is estimated to ensure the stochasticity of the model. This representation is an approximation since the probabilities for the bigrams seen in the corpus can also be estimated using a back-off path to a unigram. However, since the seen bigram typically has higher probability than its backed-off unigram and the decoding in speech recognition generally uses the Viterbi approximation, it has no significant effect on the system performance.

The pronunciation lexicon L is constructed by taking the Kleene closure of the union of pronunciations for each word. The transducer L is generally not determinizable. This can be clearly seen in the presence of homophones (two or more words having

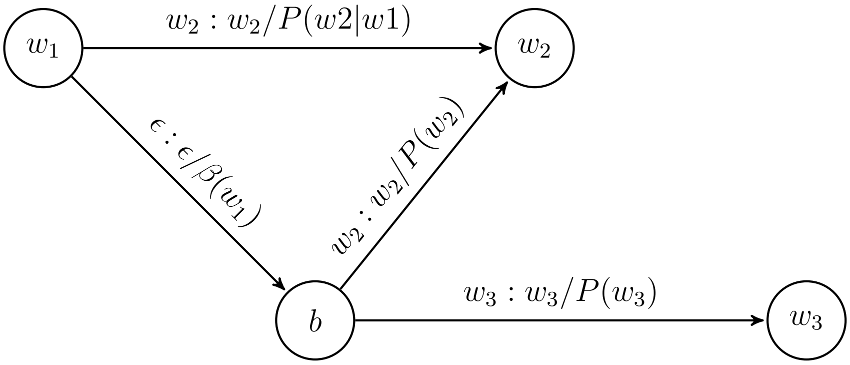


Figure 2.1. Bigram language model representation with weighted transducers or automata.

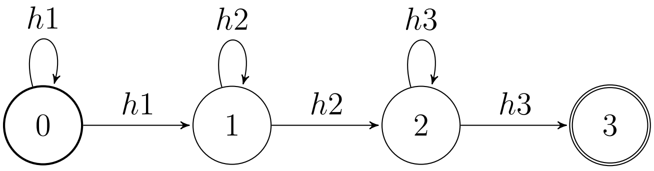
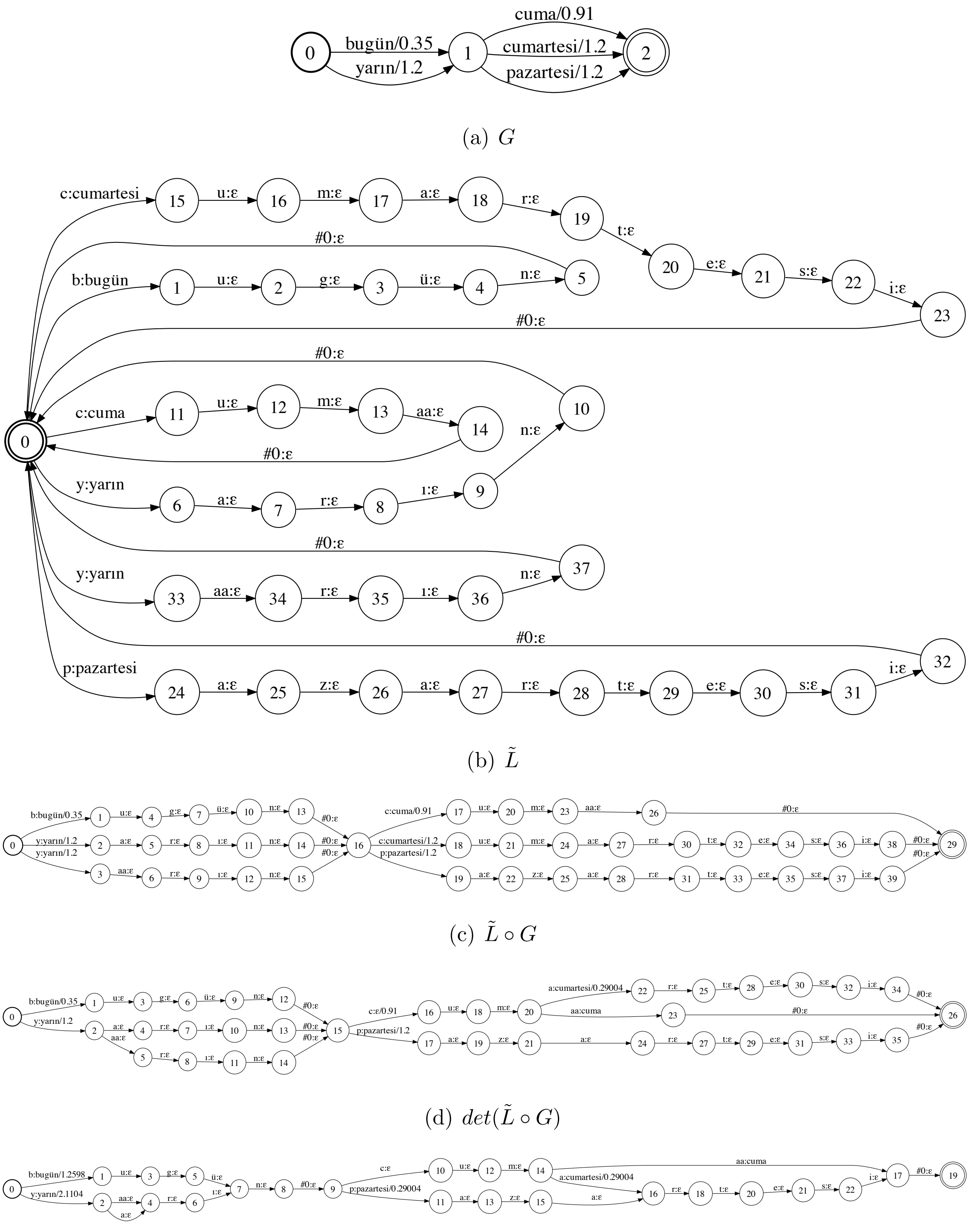


Figure 2.2. A left-to-right three-state HMM structure for a phone or triphone.

the same pronunciation). Even without the homophones, L may be non-determinizable due to unbounded ambiguity in segmenting phone strings to words. Hence, to determinize L, an auxiliary phone symbol #0 is used to mark the end of phonetic transcription of each word. Similarly, other auxiliary symbols $\#0 \ldots \#n$ are used to distinguish homophones. The lexicon transducer augmented with these auxiliary symbols is denoted by $\tilde{L}$.

A context-dependency transducer C can also be constructed to map from context-independent phones to context-dependent units like triphones [7]. In speech recognition, context dependent models are typically HMM models. A typical left-to-right HMM structure can also be represented with a finite-state transducer H mapping HMM states to HMM models as shown in Figure 2.2.

The composition algorithm matches the output label of the transitions of one transducer with the input label of the transitions of another transducer. The result is a new weighted transducer representing the relational composition of the two transducers. This operation allows us to integrate all finite-state knowledge sources into a speech recognition transducer (search network). The determinization and minimiza-



(a) G

(b) $\tilde{L}$

(c) $\tilde{L} \circ G$

(d) $det(\tilde{L} \circ G)$

(e) $min_{tropical}(det(\tilde{L} \circ G))$

Figure 2.3. (a) a simple grammar G, (b) a pronunciation lexicon $\tilde{L}$, (c) composition of lexicon and grammar transducer $\tilde{L} \circ G$, (d) determinization of the resulting transducer $det(\tilde{L} \circ G)$, (e) minimization of the determinized transducer $min_{tropical}(det(\tilde{L} \circ G))$.

tion algorithms can be used to optimize the search network to reduce decoding time and space requirements of the system. The complete set of operations for building a speech recognition transducer N can be summarized by the following formula:

$$N = \pi_{\epsilon}(min(det(\tilde{H} \circ det(\tilde{C} \circ det(\tilde{L} \circ G))))) \tag{2.1}$$

where the symbol ˜ marks the models augmented with auxiliary symbols for determinization, det is used for determinization operation, and min is used for minimization operation, and π_{ϵ} denotes the operation for replacing auxiliary symbols with ϵ. The construction of a speech recognition transducer is shown in Figure 2.3 on an example toy grammar and down to the context-independent phone level, that is $min(det(\tilde{L} \circ G))$. Figure 2.3a shows a simple grammar G and Figure 2.3b shows the pronunciation lexicon $\tilde{L}$ augmented with auxiliary symbols. Figure 2.3c shows the composition $\tilde{L} \circ G$. Figure 2.3d shows them after determinization, $det(\tilde{L} \circ G)$, which removes the non-determinism on the input side. Finally, Figure 2.3e shows the minimization of the recognition transducer over the tropical semiring, $min_{tropical}(det(\tilde{L} \circ G))$.

2.2. Discriminative Reranking: Perceptron Algorithm

In natural language processing, we can frame many tasks as a ranking or reranking problem. Discriminative ranking and reranking approaches have been proposed as an alternative to generative history-based probabilistic models [1,16–18]. In reranking tasks, a baseline generative model generates a set of candidates, and then these candidates are reranked by using local and global features, generally including the likelihood scores from the baseline model. For instance, in parsing, a baseline parser produces a set of candidate parses for each input sentence, with their associated probabilities which define an initial ranking over these parses. Then, a reranking model can be used to rerank these parses with the anticipation of improving the initial rankings using additional features derived from the parse tree. As an advantage over generative models, the discriminative reranking models allow the use of arbitrary features as evidence without concerning about the interaction of features or the construction of a generative model using these features.

```
input set of training examples ${(x_i, y_i) : 1 \leq i \leq N}$
input number of iterations $T$
$\vec{\alpha} = 0$, $\vec{\gamma} = 0$
for $t = 1 \ldots T$, $i = 1 \ldots N$ do
    $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \vec{\alpha}$
    if $z_i \neq y_i$ then
        $\vec{\alpha} = \vec{\alpha} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
    end if
    $\vec{\gamma} = \vec{\gamma} + \vec{\alpha}$
end for
return  $\vec{\gamma} = \vec{\gamma}/(nT)$
```

Figure 2.4. The averaged perceptron algorithm.

As a discriminative reranking approach, the variants of the perceptron algorithm proved to be very successful. The perceptron is a simple artificial neural network which can be used as a binary linear classifier [19]. A variant of the perceptron algorithm, the voted perceptron has been applied to classification tasks in natural language processing [20]. Another variant of the algorithm, the averaged perceptron has been shown to outperform *Maximum Entropy (Max-Ent or ME) Models* [21] in part-of-speech tagging and parsing tasks [1]. A variant for pairwise classification with uneven margins has been applied to the task of parse reranking and machine translation reranking [18].

Figure 2.4 shows a variant of the perceptron algorithm - the averaged perceptron [1, 20] formulated as a multiclass classifier. The algorithm estimates a parameter vector $\vec{\alpha} \in \Re^d$ using a set of training examples (x_i, y_i) for $i = 1 \ldots N$. The components of this algorithm is described next as outlined in the framework of [1]. The function **GEN** enumerates a finite set of candidates $\mathbf{GEN}(x) \subset Y$ for each possible input x. The representation $\mathbf{\Phi}$ maps each $(x, y) \in X \times Y$ to a feature vector $\mathbf{\Phi}(x, y) \in \Re^d$. The components **GEN**, Φ and $\vec{\alpha}$ define a mapping from an input x to an output $F(x, \vec{\alpha})$:

$$F(x, \vec{\alpha}) = \mathop{\arg\max}_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \vec{\alpha} \tag{2.2}$$

where $\Phi(x, y) \cdot \vec{\alpha}$ is the inner product $\sum_i \Phi_i(x, y) \cdot \alpha_i$. The learned parameter vector $\vec{\alpha}$ can be used for mapping unseen inputs $x \in X$ to outputs $y \in Y$ by searching for the best scoring output using this equation. The scores $\mathbf{\Phi}(x, z) \cdot \vec{\alpha}$ can also be used to rank the possible outputs for an input x.

The algorithm makes multiple passes (denoted by T) over the training examples. For each example, it finds the highest scoring candidate among all candidates using the current parameter values. If the highest scoring candidate is not the correct one, it updates the parameter vector α by the difference of the feature vector representation of the correct candidate and the highest scoring candidate. This way of parameter update increases the parameter values for features in the correct candidate and downweights the parameter values for features in the competitor. For the application of the model to the test examples, the algorithm calculates the “averaged parameters” since they are more robust to noisy or inseparable data [1]. The averaged parameters γ are calculated by summing the parameter values for each feature after each training example and dividing this sum by the total number of updates.

The perceptron algorithm tries to learn a weight vector that minimizes the number of misclassifications. The loss function of this algorithm can be written as follows.

$$L(\vec{\alpha}) = \sum_{i=1}^{N} [\![\vec{\alpha} \cdot \mathbf{\Phi}(x_i, z_i) - \vec{\alpha} \cdot \mathbf{\Phi}(x_i, y_i)]\!] \tag{2.3}$$

where $[\![x]\!] = 0$ if $x < 0$ and 1 otherwise. The variants of the perceptron algorithm have been proposed based on the notion of maximizing the margin [18,22]. In reranking tasks, the margin is defined as the distance between the best candidate and the rest and the reranking problem is commonly reduced to a pairwise classification problem.

2.3. Automatic Speech Recognition

Automatic speech recognition (ASR) is conversion of spoken words or utterances to text using computational methods. Although the ultimate goal of transcribing speech by any speaker in any environment as good as humans do has not been attained, the research in ASR has succeeded in producing many practical applications recently, especially for mobile platforms. One application area is in human-computer interaction, where ASR presents a natural eyes and hands free interface for command and control applications. Another application area is in telephony, where speech recognition is used in interactive voice response systems, such as banking and voice dialing applications. Final application area is dictation, which is transcription of speech uttered by a specific speaker. Medical dictation and mobile sms-email dictation are widely used example applications.

The speech recognition task can vary greatly in terms of difficulty depending on several parameters. One parameter is the vocabulary size of the task, which specifies the number of distinct words ASR system needs to recognize. The recognition problem becomes harder with the increasing vocabulary size. Large vocabularies (generally more than 20000 words) are required for many tasks, for instance, transcribing conversations or broadcast news. The required vocabulary size also depends on the type of language, for instance, morphologically complex languages tend to have a very rich vocabulary. A word in a utterance that is not in the vocabulary of the ASR system is said to be *out-of-vocabulary (OOV)* word and the recognizer makes recognition errors by choosing acoustically similar words in place of OOV words. The other parameter is related to how much fluent or natural speech input is allowed in the system. For isolated word recognition, it is expected that the words are uttered with short pauses between them. This is clearly easier than recognizing continuous speech such as natural conversational speech or read speech of broadcast news. The third parameter is the effect of channel and noise. The quality of the channel the speech is recorded and any kind of noise in the recordings greatly effect the speech recognition performance. The last parameter is the speaker characteristics and speaker dependence. It is easier to recognize speech that matches the training data in terms of dialect and accent. If the system is trained

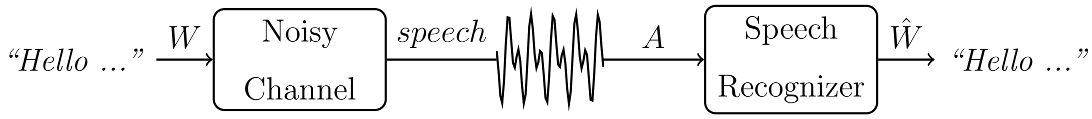


Figure 2.5. The noisy channel metaphor for ASR. The speech recognizer tries to decode the original message W which is assumed to have gone through a noisy channel to produce an acoustic waveform A.

on a specific speaker's data, the system is said to be speaker-dependent. Speaker-independent systems are harder to build and in general perform worse than speaker-dependent systems.

2.3.1. ASR Architecture

The speech recognition task we focus on in this thesis is referred as *large-vocabulary continuous speech recognition (LVCSR)*. The state-of-the-art LVCSR systems commonly use the *hidden Markov model (HMM)* paradigm. HMM-based systems model the speech recognition problem of mapping an acoustic waveform to a word sequence using the *noisy channel metaphor* shown in Figure 2.5 [23, 24]. In this metaphor, the acoustic waveform A is considered as a noisy version of a sentence (a word sequence) W that has gone through a noisy channel and the speech recognizer tries to recover the original sentence by building a model of the channel. The intuition is that we can run all the possible sentences in the language through the model of noisy channel and select the sentence whose output matches best with the acoustic waveform of the original sentence.

The probabilistic implementation of the noisy channel model is a special case of *Bayesian inference*, where the problem can be stated as finding the most likely word sequence out of all possible word sequences in the language $\mathcal{L}$ given the acoustic input A. ASR systems process the acoustic signal to produce a sequence of symbols or observations which makes it possible to estimate a probabilistic model for matching the acoustic signals. The front end acoustic processor component of a speech recognizer is responsible for this conversion. *Mel Frequency Cepstral Coefficients (MFCC)* are frequency-based acoustic features commonly used to represent acoustic observation sequences. In this representation, each observation a_i of a time interval i which corresponds to i^{th} time slice of for instance 10 milliseconds, is an n-dimensional feature vector containing MFCC parameters together with their Δ and $\Delta\Delta$ coefficients which are first and second order time derivatives of MFCC parameters. Hence, the acoustic observation sequence can be written as consecutive observation symbols:

$$A = a_1, a_2, \ldots, a_t \tag{2.4}$$

Similarly, it is convenient to represent a sentence as a sequence of words:

$$W = w_1, w_2, \ldots, w_n \tag{2.5}$$

While this simplifying assumption works well with languages such as English, it may be more suitable to use finer divisions or subtle representations depending on language characteristics. For example, a central theme of this thesis is using morphological units of words for representing Turkish sentences, where a morpheme constitutes a meaningful morphological unit of the language that cannot be further divided.

Using these assumptions, the speech recognition problem can then be expressed in the probabilistic framework as follows:

$$\hat{W} = \underset{W \in \mathcal{L}}{\arg\max}\, P(W|A) \tag{2.6}$$

The noisy channel metaphor instructs us to rewrite this equation in a different form using Bayes' rule:

$$\hat{W} = \underset{W \in \mathcal{L}}{\arg\max}\, \frac{P(A|W)P(W)}{P(A)} \tag{2.7}$$

In this equation, the probability of the acoustic observation sequence, $P(A)$ is hard to estimate. But we don't need to estimate this probability since we are searching for a sentence maximizing $P(W|A)$ and $P(A)$ is the same for each possible sentence. Thus, we can ignore $P(A)$ and simplify the equation as follows:

$$\hat{W} = \arg\max_{W\in\mathcal{L}} \frac{P(A|W)P(W)}{P(A)} = \arg\max_{W\in\mathcal{L}} \overbrace{P(A|W)}^{\textit{likelihood}}\overbrace{P(W)}^{\textit{prior}} \tag{2.8}$$

In this equation, we have two probabilities that we need to estimate. The likelihood $P(A|W)$ is called the acoustic likelihood and can be estimated using the *acoustic model (AM)*. HMMs are commonly used for acoustic modeling. The prior probability $P(W)$ is the probability of the word sequence and it is computed by the *language model (LM)*. n-grams have been dominant approach for language modeling. The operation arg max implies search for the most probable sentence maximizing the product of the AM and LM probability and this process is called decoding. The speech recognition systems commonly use a dynamic programming algorithm called Viterbi decoding. The following sections further describe these models and the decoding algorithm.

2.3.2. Acoustic Models: HMMs

A hidden Markov model is a statistical Markov model with latent states [25]. HMMs have been successfully applied to speech recognition and other sequence labeling tasks such as part-of-speech tagging. An HMM model is a finite-state machine defined by a set of parameters θ:

- A set of states $Q = q_1, q_2, \ldots, q_N$ where N is the number of states in the model.
- A set of transition probabilities $A = \{a_{ij} : 1 \leq i, j \leq N\}$ where a_{ij} is a transition probability from state i to state j. The transition probabilities out of a state must sum to 1.
- A set of observation likelihoods $B = \{b_i(o_t) : 1 \leq i \leq N, 1 \leq t \leq T\}$ where T is the number of observations and $b_i(o_t)$ is the probability of observing the symbol o_t at a state i.

Using these model parameters, an HMM can be used to estimate the probability of an observation sequence $O = o_1, o_2, \ldots, o_T$ by summing the path probabilities over

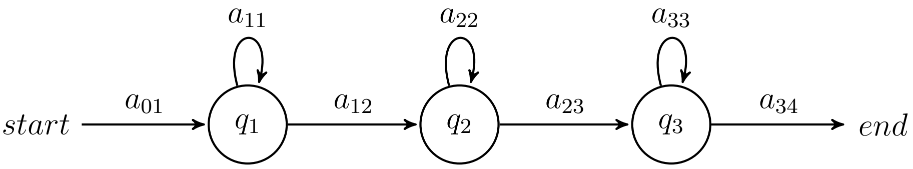


Figure 2.6. A 3-state left-to-right phone HMM with state transition probabilities.

all possible state sequences $q_1, \ldots, q_T$ generating the observation sequence as follows:

$$P(O|\theta) = P(o_1, o_2, \ldots, o_T|\theta) = \sum_{q_1,\ldots,q_T} \prod_{i=1}^{T} a_{q_{i-1},q_i} b_{q_i}(o_t) \tag{2.9}$$

Calculating this summation over all state sequences is not feasible since the number of state sequences increases exponentially with the number of observations. But, there is a simple dynamic programming algorithm called the *forward algorithm* that efficiently calculates the probability of observation sequences. Similar to the forward algorithm, there exists an algorithm which can find the best hidden state sequence for a given acoustic observation sequence. This algorithm is explained in the next section. The parameters of an HMM, namely transition probabilities and observation likelihoods, can be automatically learned using the *forward-backward* or *Baum-Welch* algorithm [26] which is a special case of the *Expectation-Maximization (EM)* [27] algorithm. This algorithm finds the local maximum likelihood estimate of the parameters of the HMM given the set of observation sequences where the probability of the observation sequence given the model $P(O|\theta)$ is locally maximized. There is also an efficient approximation to the Baum-Welch algorithm called *Viterbi training*.

In LVCSR systems, the phones are the basic acoustic modeling units which are commonly modeled by HMMs. But instead of having a model for each phone, triphones are widely used for better modeling of acoustic variations of phones due to coarticulation. A triphone also considers the left and right context of phone. For instance, a triphone HMM model for the phone ə in the pronunciation of word *run* /rən/ can be shown as r-ə+n. This representation also specifies the phones in the immediate left and right context of the phone separated by "-" and "+" symbols, respectively. HMMs for triphones are generally 3-state left-to-right models (see Figure 2.6 for an example). Since phones are basic acoustic modeling units, we need a pronunciation lexicon listing the pronunciations of each word in the language vocabulary. HMMs for words are constructed simply by concatenating triphone HMMs using the pronunciation lexicon. Training HMM-based acoustic models for LVCSR systems requires large amount of acoustic data with their sentence-level transcription. Mostly, the amount of acoustic data required for training accurate and robust models is not enough due to the large number of parameters in the model. Parameter tying which means using the same parameters several times in the model is a frequently used approach in training. For estimating the observation likelihoods at each HMM state, *Gaussian Mixture Models* are often used where each state is modeled as a weighted mixture of a number of gaussians. Parameter tying can be both at the HMM state level and at the level of gaussians.

2.3.3. Speech Decoding: Viterbi Algorithm

In speech recognition, the problem of finding the most probable word sequence given an acoustic observation sequence is a search problem called speech decoding. In HMM-based systems, a dynamic programming algorithm called *Viterbi algorithm* [28] is commonly used. The Viterbi algorithm operates on a finite-state machine such as an HMM.

The algorithm as applied to decoding the best state sequence (path) of an HMM state machine whose parameters are defined in the previous section for a given observation sequence is given in Figure 2.7. In this algorithm, $v_t(s)$ is the Viterbi path probability which expresses the probability of the most likely state sequence that is in state s after seeing the first t observations. The algorithm chooses the path probability maximizing the product of the best Viterbi path probability from the previous time step $v_{t-1}(s')$ of each state s' and the transition probability $a_{s's}$ between the states s' and s. It also stores the state number of the previous path that leads to the best path probability for the current time step for each state in *back-pointer*$[s][t]$. This is used for backtracing the most likely state sequence starting from the last state of the best path, $s = \arg\max_{1 \leq s' \leq N} v_{s'}(T)$.

```
input observation sequence $O = o_1, o_2, \ldots, o_T$
input graph with states $Q = q_1, q_2, \ldots, q_N$
input state transition probabilities $A = \{a_{ij} : 1 \leq i, j \leq N\}$
input observation likelihoods $B = \{b_i(o_t) : 1 \leq i \leq N, 1 \leq t \leq T\}$
$v_0(0) = 1.0$
for $t = 1 \ldots T$ do
  for $s = 1 \ldots N$ do
    $v_t(s) = \max_{1 \leq s' \leq N} [v_{t-1}(s') * a_{s's}] * b_s(o_t)$
    back-pointer$[s][t] = \arg\max_{1 \leq s' \leq N} [v_{s'}(t-1) * a_{s's}]$
  end for
end for
return the optimal state sequence obtained by backtracing starting
from state $s = \arg\max_{1 \leq s' \leq N} v_{s'}(T)$
```

Figure 2.7. The Viterbi algorithm for speech decoding.

While the Viterbi algorithm is much more efficient than enumerating all the possible state sequences and calculating the probabilities for them, it is still slow $(O(N^2T))$ for speech decoding since there can be a large number of states in the speech decoding network that needs to be considered for each time step. Therefore, an approximation of the Viterbi algorithm called the *beam search* is commonly used. In beam search, an active list of states with the Viterbi path probabilities are kept for each time step. In the next time step, we extend only the transitions out of the states having the path probability within a fixed threshold of the best path probability of the current active states. The other states are pruned away since they represent the low-probability unpromising paths. But if it turns out that one of the pruned paths would be in fact the prefix of the final best path, we make a *search error* by using the beam search approximation. Hence, there is a trade-off between speed-up (aggressive pruning) and recognition performance. Besides, the Viterbi algorithm in speech decoding does not actually compute the word sequence which is most probable given the acoustic input. Instead, it is decoding the best state or phone sequence. This is called *Viterbi approximation*, since it is approximating the best word sequence probability by calculating the best state sequence probability rather than summing over all possible state sequences

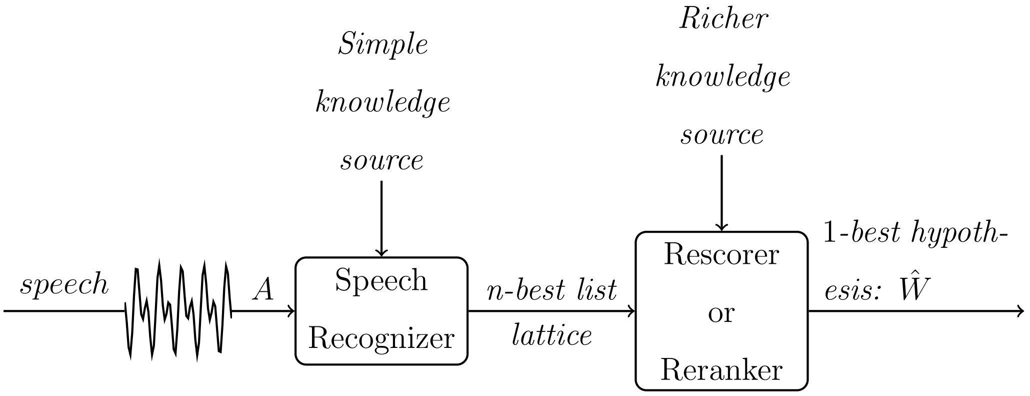

Simple knowledge source
Richer knowledge source
speech
A
Speech Recognizer
n-best list
lattice
Rescorer or Reranker
1-*best hypothesis:* $\hat{W}$

Figure 2.8. The rescoring or reranking of ASR hypotheses represented as n-best lists or word lattices.

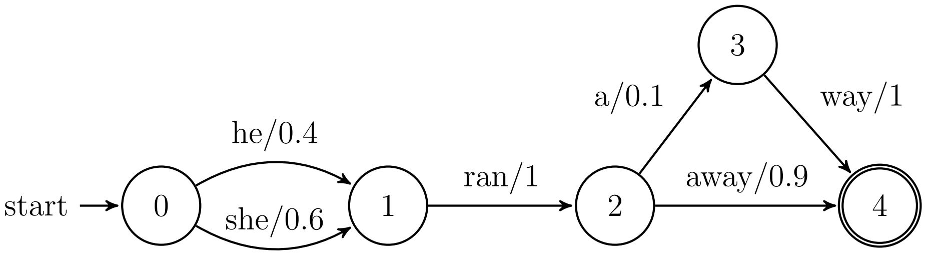


Figure 2.9. A word lattice example representing word hypotheses with word probabilities.

that can generate a word sequence. It turns out that this is mostly a reasonable approximation. However, it may be disadvantageous for decoding of words having multiple pronunciations or having multiple analyses due to morphological ambiguity.

The speech recognition system are evaluated using the *word error rate (WER)* metric. The calculation of WER uses the computation of *minimum edit distance* in words between the hypothesized word string and the correct or *reference* transcription. The WER is then defined as the minimum number of word substitutions, word insertions, and word deletions necessary to map between the correct and hypothesized strings divided by the total number of words in the reference transcription and it is given as a percentage as follows:

$$\text{WER} = 100 \times \frac{\text{substitutions} + \text{insertions} + \text{deletions}}{\text{total words in correct transcript}} \tag{2.10}$$

A cause for recognition errors is the use of approximate or inaccurate models in the decoding. In speech decoding, it is often the case that the search space is very large and using more accurate and sophisticated models is not efficient and feasible. Therefore, the decoding process is generally carried out in two stages called *multiple-pass decoding*. In the first pass, we use time and space efficient knowledge sources or algorithms to generate a list of hypotheses rather than generating 1-best hypothesis (i.e. the best path in Viterbi decoding). In the second pass, we can use richer and more sophisticated models or algorithms for decoding in a reduced search space defined by the hypotheses generated in the first pass. Hence the second pass can be considered as *rescoring* or *reranking* the candidate hypotheses as shown in Figure 2.8. The intermediate hypotheses can be represented as an *n-best list* or *word lattice*. An n-best list is a list of hypotheses with the best scores from the first pass decoding. A word lattice is a directed graph that compactly represents the hypotheses of word sequences with possibly more information such as word probabilities and timing information. An example word lattice is shown in Figure 2.9. There are a number of algorithms augmenting the Viterbi algorithm to generate n-best hypotheses or word lattices [2, 29]. The recognition error rate from the second pass has a lower bound determined by the n-best or *lattice error rate* which is the word error rate we get by choosing the hypothesis with the lowest number of errors for each n-best list or lattice. It is also called *oracle error rate* since it requires perfect knowledge of correct choice in each case.

2.3.4. Generative n-gram Language Models

A statistical *language model (LM)* assigns a probability to a sentence by estimating a probability distribution over word sequences. Language modeling is an essential tool for many speech and language processing tasks. For instance, the speech recognition problem as formulated by the noisy channel model requires an estimation of a prior probability $P(W)$ over word sequences, which can be used to predict the next word in a noisy input. In machine translation, they are used for assigning a probability for each possible sentence in the target language to decode a well-formed and probable translation. In spelling correction, they can be used to accomplish context sensitive spelling corrections.

In speech and language processing, a statistical language modeling technique called *n-gram language modeling* has been tremendously successful. The probabilistic language models, n-grams, formalize the idea of predicting the next word in a word sequence by using the previous $n-1$ words. Below, for the formulation of n-grams, we write the probability of a word sequence as conditional probabilities using the chain rule.

$$P(W) = P(w_1, w_2, ..., w_N) = \prod_{i=1}^{N} P(w_i|w_1, w_2, \ldots, w_{i-1}) \tag{2.11}$$

The conditional probabilities in this equation are word probabilities conditioned on preceding words or also called history of words. It is very hard to estimate these probabilities reliably, since the word histories can be very long and the number of parameters exponentially increases with the length of history which leads to data sparsity problem for statistical estimation of the parameters. Hence, n-grams makes an approximation as follows:

$$P(W) = P(w_1, w_2, ..., w_N) \approx \prod_{i=1}^{N} P(w_i|w_{i-n+1}, \ldots, w_{i-1}) \tag{2.12}$$

Here, it is assumed that the probability for the i^{th} word w_i given the preceding $i-1$ words can be approximated by the probability of it given the preceding $n-1$ words, i.e. we are limiting the history to $n-1$ words.

The parameters of n-grams can be estimated from a text corpus using the *Maximum Likelihood Estimation (MLE)*. This method gives an estimation for the conditional probabilities using the relative frequency counts of n-grams (i.e. a particular word and its history) and their histories as follows:

$$P(w_i|w_{i-n+1}, \ldots, w_{i-1}) = \frac{count(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{count(w_{i-n+1}, \ldots, w_{i-1})} \tag{2.13}$$

where $count(w)$ gives the number of occurrences of the word string w in the training text data.

An n-gram language model is called a unigram, bigram or trigram language model as a Markov model of n^{th} order 1, 2, or 3 respectively. The number of parameters in the n-gram model increases exponentially with the order of the model given a vocabulary size $|\mathcal{V}|$, i.e. $|\mathcal{V}|^n$. Given a limited amount of text data even if it is a large amount, increasing n-gram order may quickly result in nonrobust parameter estimations due to data sparsity problem. On the other hand, using higher order n-grams generally increases the prediction power of the model given enough amount of training data. Hence, there is a trade-off between n-gram order and robust parameter estimation. There is also time and space efficiency factor for language models limiting the use of higher order n-grams. As a consequence, in speech and language processing, 3-grams have proved to be a good trade-off. But, it has been a common approach to use higher order language models such as 4-grams in the second pass of a multipass system. For instance, in speech recognition, rescoring or reranking hypotheses in a second pass with higher order n-grams than ones used in the first pass is often applied. The size of n-gram language models can also be reduced for space and time efficiency. An entropy-based pruning technique based on a criterion of relative entropy between the original and the pruned model is generally used for reducing the model size without degrading the model quality much [30].

The estimation of model parameters using the relative frequencies has one major problem due to data sparsity. Some of the n-grams may be missing in the training data, therefore the language model assigns zero probability to sentences having n-grams not seen in the training data. Another problem is that MLE underestimates the probability estimates for n-grams that occur very infrequently in the training corpus. Therefore, in n-gram language modeling, smoothing techniques are commonly applied to overcome these problems. Smoothing techniques reserve some probability mass from frequent n-grams and distribute this mass over zero count or infrequent n-grams. There are also two other common ways for smoothing, back-off and interpolation. In back-off, we back off to a lower order n-gram model for probability estimation if we have zero evidence for a higher order n-gram. By contrast, in interpolation, we always do a weighted interpolation of lower and higher order n-grams by mixing the probability estimates from all the n-gram estimators. The *Interpolated Kneser-Ney algorithm* is the most commonly used n-gram smoothing technique [31]. A comparison of several smoothing techniques can be found in [32]. A review of statistical language modeling techniques including n-grams and *Maximum Entropy Language Modeling* can be found in [33].

Finally, we describe how to evaluate the language models. The correct way to evaluate the performance of a language model is to use the language model in a task and measure the system performance. For instance, in speech recognition, using different language models for speech decoding and comparing the corresponding word error rates is a standard approach. But, the repetitive application of a language model in a task can be expensive. Therefore, an evaluation metric called *perplexity* is commonly used for n-gram language models. Even if perplexity improvement does not guarantee a performance improvement in the final system integration, there is often a correlation. Nevertheless they can provide a quick check for a language modeling method before the language model is applied in a real task. The perplexity of an n-gram language model on a test set $W = w_1, w_2, ..., w_N$ is defined as the normalized probability of the test set by the number of words:

$$perplexity(W) = P(W)^{-\frac{1}{N}} = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}} \tag{2.14}$$

Using the conditional probabilities by the chain rule:

$$perplexity(W) = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}} = \left[\prod_{i=1}^{N} P(w_i|w_{i-n+1}, \ldots, w_{i-1})\right]^{-\frac{1}{N}} \tag{2.15}$$

According to this equation, maximizing the test set probability minimizes the perplexity. The perplexity can also be considered as a weighted average branching factor of a language which is defined as the average number of possible next words that can follow any word. The perplexities of two language models can only be compared if the language models have the same vocabulary.

2.3.5. Discriminative Language Models

A complementary approach to generative n-gram language models are *discriminative language models (DLMs).* DLMs are discriminatively trained models proposed for large vocabulary speech recognition and do not attempt to estimate a generative model $P(W)$ over word strings. Instead, they are trained on acoustic sequences with their transcriptions, in an attempt to directly optimize word error rate [34]. There are two parameter estimation methods commonly used for training DLMs. One of them is the perceptron algorithm which can be used to build a discriminative global linear model. And the other one is a *global conditional log-linear model (GCLM)* which is based on maximizing the regularized conditional log-likelihood. These models can be trained over n-best lists and word lattices.

Discriminative language modeling is formulated in the framework outlined in [1]. In this framework, the task is to learn a mapping from inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$. The training examples are (x_i, y_i) for $i = 1 \ldots N$. A function **GEN** enumerates a set of candidates $\mathbf{GEN}(x)$ for an input x. Φ is a representation mapping each $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a feature vector $\Phi(x, y) \in \Re^d$. The task is to learn a parameter vector $\vec{\alpha} \in \Re^d$. The components **GEN**, Φ and $\vec{\alpha}$ define a mapping from an input x to an output $F(x, \vec{\alpha})$:

$$F(x, \vec{\alpha}) = \operatorname*{arg\,max}_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \vec{\alpha} \tag{2.16}$$

where $\Phi(x, y) \cdot \vec{\alpha}$ is the inner product $\sum_i \Phi_i(x, y) \cdot \alpha_i$. The perceptron and the GCLM are the parameter estimation methods that can be used to learn the parameter vector $\vec{\alpha}$. The decoding in this framework is searching for the output y that maximizes $\Phi(x, y) \cdot \vec{\alpha}$.

GCLMs use the parameters $\vec{\alpha}$ to define a conditional distribution over the members of $\mathbf{GEN}(x)$ for a given input x:

$$P_{\vec{\alpha}}(y|x) = \frac{exp(\Phi(x, y) \cdot \vec{\alpha})}{Z(x, \vec{\alpha})} = \frac{exp(\sum_i \Phi_i(x, y) \cdot \alpha_i)}{Z(x, \vec{\alpha})} \tag{2.17}$$

where $Z(x, \vec{\alpha}) = \sum_{y \in \mathbf{GEN}(x)} exp(\Phi(x, y) \cdot \vec{\alpha})$ is a normalization constant that depends on x and $\vec{\alpha}$. The regularized log-likelihood of the training data is used as the objective function to optimize to learn the parameters $\vec{\alpha}$. Even though $\mathbf{GEN}(x)$ can be exponential in size, the GCLMs can use efficient dynamic programming algorithms for training and decoding by relying on the local nature of the feature vector representation $\Phi(x, y)$.

In this framework, discriminative language modeling for speech recognition can be applied as follows. We define $\mathcal{X}$ as the set of all possible acoustic inputs. $\mathcal{Y}$ is taken as the set of all possible word strings, Σ^* for a vocabulary Σ. Then, each x_i is an acoustic feature vector for an utterance, and $\mathbf{GEN}(x_i)$ is the set of candidate hypotheses (n-best list or lattice) output from a first pass decoding for the acoustic input x_i. The y_i is taken as the hypothesis in $\mathbf{GEN}(x_i)$ with the lowest word error rate with respect to the reference or correct transcription of x_i. With this setting, we can learn a parameter vector $\vec{\alpha}$ using one of the parameter estimation methods and use that to choose the hypothesis with the best score of $\Phi(x, y) \cdot \vec{\alpha}$.

2.4. Turkish Broadcast News Transcription

In this section, we discuss the characteristics of the Turkish language and its challenges in automatic speech recognition. Besides, we describe the broadcast news transcription system on which we carry out our experiments.

2.4.1. Turkish Language: Characteristics

Turkish is an agglutinative language with a highly productive inflectional and derivational morphology. The rich morphology leads to a large (possibly infinite) number of words in Turkish. The most common order for the syntactic constituents is Subject-Object-Verb. But Turkish has a rather flexible word order, since having a rich set of morphological markings helps to disambiguate the grammatical roles of the syntactic constituents without relying on the word order. For instance, all the six permutations of the following words "yaşa *(live)*, hayatını *(your life)*, aşkla *(with love)*" constitute a grammatically and semantically correct sentence and the choice depends on the discourse context or the speaker's will to emphasize a word.

Turkish has almost one-to-one mapping between graphemes and phonemes. The number of exceptional root words having a different pronunciation than which is directly implied by its graphemics is about 3700 according to our compilation of a pronunciation lexicon for Turkish. These exceptional root words are mostly loan words from other languages such as Arabic, Persian, and English and most of them are relatively infrequent in current usage. Hence, the 29 graphemes (8 vowels and 21 consonants) of Turkish are often taken as the phone alphabet for the acoustic modeling. Using a larger phone set has not improved the speech recognition performance in our experiments.

Turkish morphology has both derivational and inflectional suffixation. The inflectional suffixes change the form of a word to express a grammatical function or attribute such as tense, mood, person, number, case and gender without changing the grammatical category (Part-of-Speech or PoS) of the word. For instance, the inflectional suffix /m/ for the first person possessive case can be seen in the following word: kedi*(cat)*m*(my) (my cat)*. On the other hand, the derivational suffixes form a new word while possibly changing the syntactic category of the word. An example derivational suffix is /çi/ as in çiçek*(flower)*çi *(florist)*.

In the suffixation process, the stem or the suffixes may undergo some orthographic changes due to phonological phenomena like vowel and consonant harmony. Hence, there is made a distinction between the lexical representation of the stems and suffixes, and the surface form realizations of the words with suffixation. For an example, consider the words kalemler*(pencils)* and kitaplar*(books)* having kalem(pencil) and kitap(book) stem, respectively. The /ler/ and /lar/ suffixes are two possible surface realizations of the lexical plural suffix /lAr/, whose graphemic realization depends on the last vowel of the stem according to the vowel harmony rule. These type of phonological or orthographical alternations are governed by a set of rules called morphophonemics or morphographemics.

2.4.2. Turkish Language: Challenges for Speech Recognition

The productive inflectional and derivational morphology of Turkish leads to a large number of distinct words in the vocabulary of the language. The suffixation process enables to form a large number of new words from a stem. In theory, the vocabulary of Turkish is unlimited since iteration of some suffixes such as causative suffix is possible. This vocabulary growth problem presents a challenge in speech recognition where generally a fixed-size vocabulary is used for language modeling. The flexible order of the constituents in Turkish combined with its unlimited vocabulary make the problem more pronounced by leading to data sparseness problem in parameter estimation of language models.

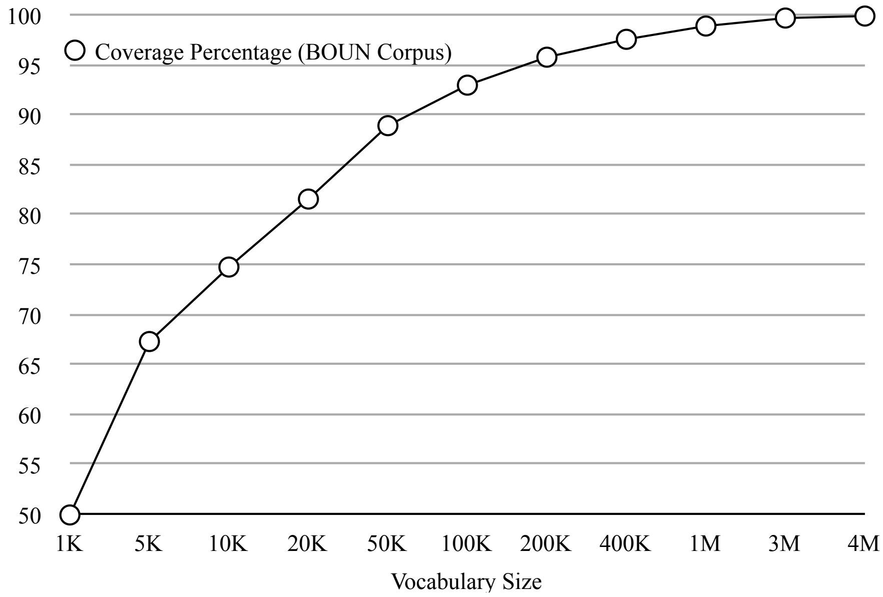


Figure 2.10. Coverage statistics for most frequent types.

The vocabulary growth problem makes it harder to obtain a fixed sized vocabulary with a good coverage. In speech recognition, the words that are not in the recognition vocabulary are called out-of-vocabulary (OOV) words. The speech recognition system can not recognize OOV words since they are not in its vocabulary. Hence, having higher OOV rates leads to lower speech recognition accuracy. It has been estimated that each OOV word in the test data results in at least 1.5 recognition errors on average [35]. Figure 2.10 shows the coverage rate for different vocabulary sizes (the topmost frequent word types) on a Turkish web corpus (BOUN Corpus). For a comparison between a morphologically productive language Turkish and a morphologically unproductive language English, an optimized 60K vocabulary for English has less than 1.0% OOV rate on North American Business news [36], while the same vocabulary size for Turkish results in about 10% OOV rate on a web corpus collected from online news portals.

For solving the OOV and data sparsity problem for Turkish speech recognition, we propose using grammatical sub-lexical units in this thesis. Using the grammatical units of lexical morphemes for vocabulary in language modeling requires solutions for two problems. The first problem is obtaining the lexical morphemes from the words. The extraction of morphological information hidden in the structure of words calls for morphological parsing, which is the decomposition of words into constituent morphemes and associated morphosyntactic and morphosemantic features. An example morphological analysis for the word *ölümsüzleştirilebileceğini* is shown below:

> *ölüm*[Noun]+[A3sg]+[Pnon]+[Nom]-*sHz*[Adj+Without]
> -*lAş*[Verb+Become]-*DHr*[Verb+Caus]-*Hl*[Verb+Pass]
> -*YAbil*[Verb+Able]+[Pos]-*YAcAk*[Noun+FutPart]+[A3sg]
> +*SH*[P3sg]+*NH*[Acc]

This word can be translated as *"... that s/he can be immortalized"*. The morphological feature representation is similar to the one used in [37]. Each output of the morphotactics begins with the root word and its part-of-speech tag in brackets. These are followed by a set of lexical morphemes associated with morphological features (nominal features such as case, person, and number agreement; verbal features such as tense, aspect, modality, and voice information). The inflectional morphemes start with a + sign. The derivational morphemes start with a - sign and the first feature of a derivational morpheme is the part-of-speech of the derived word form.

The second problem is that the morphological parser may return more than one possible analysis for a word due to ambiguity. This parsing ambiguity needs to be resolved for further language processing such as for language modeling using a morphological disambiguator (morphosyntactic tagger). For example, the parser outputs four different analyses for the word *kedileri* as shown below. The English glosses are

Table 2.1. Statistics for the NewsCor corpus.

# of word tokens	182622247
OOV rate (word token)	1.3
# of word types	1819157
OOV rate (word type)	38.8
average # of parses per word type	2.4
average # of morphemes per word type	3.7
root with max # of parses (3545)	çık[Verb]
word with max # of morphemes (9)	ruhsatlandırılamamasındaki

given in parentheses.

kedi[Noun]+*lAr*[A3pl]+*SH*[P3sg]+[Nom] (his/her cats)
kedi[Noun]+*lAr*[A3pl]+[Pnon]+*YH*[Acc] (the cats)
kedi[Noun]+*lAr*[A3pl]+*SH*[P3pl]+[Nom] (their cats)
kedi[Noun]+[A3sg]+*lArH*[P3pl]+[Nom] (their cat)

The statistics for the number of tokens (words and lexical units such as punctuation marks), types (distinct tokens), and morphological parsing are shown in Table 2.1. There are 3.7 morphemes per word type on the average. The distribution for the number of morphemes on the NewsCor corpus can be seen in Figure 2.11.

2.4.3. System Description

We apply the proposed methods of this thesis on a broadcast news transcription system for Turkish. This system has been constructed by Arısoy *et al.* [3, 38]. The acoustic models of the system have been trained using the acoustic corpus of Broadcast News (BN) database [3, 38]. It contains the broadcast news recordings from a radio channel (VOA) and four TV channels (CNN Türk, NTV, TRT1, TRT2), and their corresponding transcriptions. To obtain the transcripts, the speech recordings have been manually transcribed. The continuous speech recordings have been first automatically segmented into smaller pieces using acoustic features and then have been manually checked for incorrect segmentations. This segmentation corresponds to an

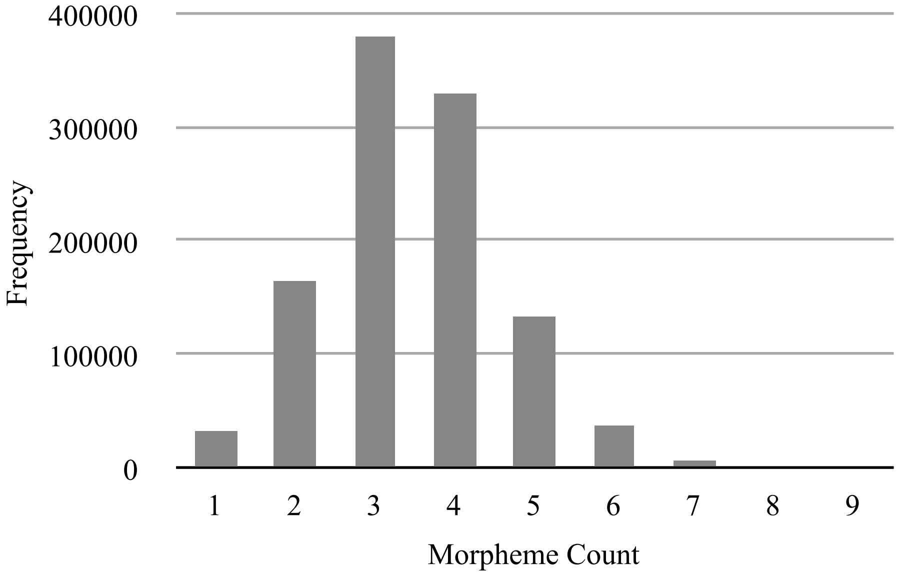


Figure 2.11. The histogram for the frequency of a specific number of morphemes in a word.

initial acoustic segmentation. The resulting acoustic segments are then further segmented into linguistic segments using the punctuations in the reference transcripts. This linguistic segmentation has been obtained automatically by force-aligning a reference transcript with its acoustic segment as explained in [3]. For the speech recognition experiments, we used about 194 hours of speech data from the BN database. Table 2.2 shows the breakdown of this data according to its partitioning and its acoustic conditions. The training data partition constitutes 188 hours of speech data and it has been used for training the acoustic models of the first-pass [3, 38] used in this thesis and for training the discriminative models in this thesis. The remaining speech data is partitioned to a held-out set of 3.1 hours and a test set of 3.3 hours of speech data. The transcriptions of the training set contain about 1.3 million words while the held-out set contains 23199 words and the test-set contains 23410 words. The automatic transcription system uses hidden Markov models (HMMs) for acoustic modeling. The HMMs are decision-tree state clustered cross-word triphone models with 10843 HMM states and each state is a Gaussian mixture model (GMM) having 11 mixture Gaussian densities with the exception of silence model having 23 mixtures.

The language models of this work were trained using two text corpora. The

Table 2.2. Partitioning of data for various acoustic conditions from [3]: f_0 is clean speech, f_1 is spontaneous speech, f_2 is telephone speech, f_3 is background music, f_4 is degraded acoustic conditions, and f_x is other.

Partition	f_0	f_1	f_2	f_3	f_4	f_x	Total (hours)	Words
Train	67.2	15.7	8.3	19.8	73.6	3.3	188	1.3M
Held-out	1.1	0.1	0.1	0.5	1.3	0.0	3.1	23199
Test	0.9	0.1	0.1	0.7	1.4	0.1	3.3	23410

smaller one is referred as the BN text corpus (1.3 million words) which contains the reference transcriptions of BN training dataset and acts as in-domain data. The larger corpus is the NewsCor corpus (184 million words) described in Section 3.3 and acts as a generic corpus collected from news portals. The generative language models in this thesis are built by linearly interpolating the language models trained on these corpora. The interpolation constant is chosen to optimize the perplexity of held-out set. The n-gram language models are estimated with interpolated Kneser-Ney smoothing and entropy-based pruning [30] using the SRILM toolkit [39]. The discriminative models are trained using only the BN corpus. We use WFSTs for model representation and model construction. The speech recognition experiments are performed by using the AT&T DCD library [2] . This library is also used for the composition and optimization of the finite-state models, which results in a search network for decoding.

The first-pass recognition experiments in this thesis are carried out on the linguistic segmentations of the acoustic data. But, for the rescoring experiments where the candidate hypotheses from the first-pass are rescored with an unpruned language model or reranked with a discriminative language model, we concatenate the hypotheses for the consecutive segments belonging to the same sentence as explained in [3]. Since we are using the same system with Arısoy [3], we repeat Arısoy's experimental results for the baseline system in Figure 2.3. These results are for the BN transcription system using a 200K vocabulary size word language model, which has about 2% OOV rate on the test set. First three results show the effect of linear interpolation for the language models using the in-domain BN corpus and general NewsCor corpus. The

[2] http://www.research.att.com/~fsmtools/dcd/

Table 2.3. Baseline broadcast news transcription results from Arısoy's work [3].

Language Model	WER (%)
generic (NewsCor) + in-domain (BN) LMs	23.4
generic (NewsCor) LM	25.2
in-domain (BN) LM	31.2
generic (NewsCor) + in-domain (BN) LMs without OOV words	22.3
generic (NewsCor) + in-domain (BN) + test data (cheating) LMs	14.9

last two results are part of a cheating experiment designed to demonstrate the lower bounds for the WER of the system. In the first experiment, all the words in the test set is included in the recognition vocabulary of the system to ensure there is no OOV word. The second experiment simulates the condition in which every sentence of the test set has already been seen, which is accomplished by using the transcripts of the test set in the language model estimation.

2.5. Related Work

In this thesis, we focus on solving the problems associated with the rich morphology of Turkish and other similar languages in a setting for a large vocabulary speech recognition task. This section reviews the previous work proposed for solving similar problems in morphologically complex languages in various speech and language processing tasks. This will help to evaluate the contributions of this thesis and to compare and contrast the proposed methods with the previous approaches.

The issues and challenges stated in the previous section are not problems specific to Turkish. The OOV and data sparsity problems are common for morphologically productive languages such as Arabic, Czech, Finnish and Korean in addition to Turkish. Hence, there have been a large number of studies to solve these problems in various speech and language processing tasks, such as speech recognition, machine translation, part-of-speech tagging and spelling correction.

Sub-lexical approaches aim to solve the mentioned problems by using grammatical or statistical sub-lexical units as the basic unit of the language instead of commonly used words. The sub-lexical units are effective in reducing OOV rate while at the same time decreasing the vocabulary size and alleviating data sparsity problem by using less number of parameters and training with higher number of statistics from data. The grammatical sub-lexical units can be morphological units such as morphemes or some grouping of them such as stem and ending (grouping of suffixes). The statistical sub-lexical units can be obtained by splitting words using statistical methods.

A morpheme-based n-gram language model has been shown to improve the perplexity values over conventional word n-gram models for German [40] which has many word inflections and compound words. But, the improvement in the speech recognition performance has been insignificant due to more confusable acoustic modeling units of morphemes. Similarly, a morpheme-based language model has been compared with a word-based model in an LVCSR system for highly inflectional Czech language [41]. However, the morpheme-based system did not have performance improvement over the word-based system in the first-pass recognition. The recognition units of merged morphemes have been proposed for a broadcast news transcription task in Korean which is an agglutinative language [42]. The merging of frequent and short morpheme pairs aimed to solve the inter-morpheme coarticulation problem when morphemes are used as recognition units. The proposed method reduced the word error rate of the baseline system (the system without morpheme merging) by 3.4% absolute. Morpheme-based language modeling has also been used for Arabic LVCSR [43]. The morpheme language model yielded an absolute improvement of 2.4% with a medium vocabulary size (64K) and only 0.2% with a large vocabulary size (800K). In this study, a morpheme lattice constrainer was used for constraining the morpheme lattices with a finite-state acceptor to solve the problem of decoding illegal morpheme sequences which results in outputs that are non-words. The stems and endings have also been used as sub-lexical units in language modeling of an inflected language Slovenian [44]. A new search algorithm has also been proposed in this work to restrict the decoding to the correct order of sub-word units, and hence the search space is constrained for efficiency.

Sub-lexical units can also be obtained by statistical methods in contrast to using a rule-based morphological parser to get grammatical sub-lexical units. The statistical methods aim to obtain a segmentation of words by learning morphology of a language in an unsupervised manner. A method for unsupervised acquisition of morphology in European languages using the minimum description length (MDL) analysis has been proposed in [45]. For unsupervised discovery of morphemes, an algorithm that is better suited for highly inflectional and agglutinative languages has been proposed [46]. The so-called Morfessor algorithm also uses the MDL principle for unsupervised segmentation of words into morpheme like units called *morphs*. Statistical sub-lexical units or morphs obtained with the Morfessor algorithm have been used for language modeling in many languages including Finnish [47, 48] and Turkish [3, 38].

Sub-lexical language modeling has also been extensively studied for Turkish. Morpheme-based language models were proposed for Turkish LVCSR in [49]. However, due to morphological ambiguity problem, syllables were utilized instead of grammatical morphemes as language modeling units. A statistical language model based on morphological decomposition of words into roots and inflectional groups has been proposed in [50]. The inflectional groups contain the inflectional features for each derived form. The roots and inflectional groups are the language modeling units. The proposed models have been used for morphological disambiguation, spelling correction, and n-best list rescoring for speech recognition. A comparative study of morpheme, stem+ending and syllable language models has been presented in [51], however, the speech recognition experiments were carried out on a small vocabulary isolated word recognition task. This work has been extended to continuous speech recognition with a new model combining words, stem+endings, and morphemes in [52]. Statistical sub-lexical units obtained with the Morfessor algorithm were shown to outperform words and grammatical units for Turkish first in [53]. The language models of this study were trained on a text corpus of 2 million words. A more recent study investigated using words, stem+endings and syllables as language modeling units on an LVCSR task using 34 hours of acoustic data and 81 million-word text corpus [54]. The best performance was obtained using stem+ending model. The further improvements in recognition accuracy were attained by incorporating language constraints addressing the vowel harmony. The language constraints are represented with a rule-based weighted finite-state machine and they are applied by composing with the lattice output from the first-pass in a second-pass lattice rescoring. The most comprehensive study on grammatical and statistical sub-lexical units for language modeling of Turkish has been done recently in [3, 38]. The sub-lexical units have been shown to outperform word-based models on a broadcast news transcription system. Arısoy [3, 55] has also addressed the over-generation problem of sub-lexical units by dynamic vocabulary adaptation, which further improves the accuracy of sub-lexical units. Moreover, discriminative language models with linguistically and statistically motivated features have been extensively studied in [3] with further performance improvements over generative models.

Sub-lexical language models alleviate the OOV problem, however the speech decoder can generate ungrammatical sub-word sequences and post-processing of the sub-word lattices may be required to correct the errors and increase the accuracy [43,54,55].

Morphology-based language modeling approaches specifically *Factored Language Models (FLMs)* have been shown to reduce language model perplexity and lead to WER reductions in Arabic speech recognition systems [56]. FLMs decompose words into a set of features (or factors) and estimate a language model over these factors, smoothed with *generalized parallel backoff* mechanism which improves the robustness of probability estimates for rarely observed n-grams.

Morphological information can also be employed later in the system as in [57,58], where a maximum entropy model has been trained with morphological and lexical features to rescore n-best hypotheses for Arabic speech recognition and machine translation.

3. TURKISH LANGUAGE RESOURCES

Turkish is an agglutinative language with a highly productive inflectional and derivational morphology which is quite regular [59]. Language applications for morphologically rich languages often require to exploit the syntactic and semantic information stored in the word structure. Therefore, we need some language resources and tools to extract and utilize this information.

There are some previous computational studies on Turkish morphology. Oflazer [60] gives a two-level morphological description implemented using the PC-KIMMO environment [61]. However, its lexicon coverage is quite limited and it requires the PC-KIMMO system to run the parser which prevents the integration of the parser into other applications. Later, Oflazer has reimplemented this specification using Xerox finite-state tools,[3] *twolc* (a two-level rule compiler) [62] and *lexc* (a lexicon compiler). This implementation requires the Xerox software for execution and the parser is not publicly available. Öztaner [63] also uses Xerox tools to build a morphological parser. Güngör [64] describes Turkish morphophonemics and morphotactics using Augmented Transition Network formalism. Despite these studies, there is no publicly available state-of-the-art morphological parser for Turkish. Considering the success of finite-state machines in language and speech processing [5], it is essential for a Turkish morphological parser to be available as a finite-state transducer in order to incorporate the morphology of the language as a knowledge source into other finite-state models.

A morphological parser may return more than one possible analysis for a word. This parsing ambiguity needs to be resolved for further language processing using a morphological disambiguator (morphosyntactic tagger). There are several studies for morphosyntactic tagging in morphologically complex languages such as Czech [65], which is an inflective language, and Basque [66] and Hungarian [67], which are agglutinative languages. For morphological disambiguation in Turkish, several constraint-based methods have been applied [68,69], where constraint voting is the primary mech-

[3] Personal communication.

anism for parse selection. A statistical model has also been used [70], where statistics over inflectional groups (chunks formed by splitting the morphological analysis of a word at derivation boundaries) that include features for intermediate derived forms are estimated by a trigram model. A recent work has employed a decision list induction algorithm called Greedy Prepend Algorithm (GPA) to learn morphological disambiguation rules for Turkish [71]. In this work, a separate decision list-based model is trained for each of the 126 morphological features and they are used to vote on the potential parses of a word. The voted or averaged perceptron algorithms that have been previously applied to classification problems [20] have also been adapted very successfully to common natural language processing (NLP) tasks such as syntactic parsing of English text [16] and part-of-speech tagging and noun phrase chunking [1]. This methodology was also proved to be quite successful for morphological disambiguation of Turkish text [72].

Due to the productive morphology and parsing ambiguity in agglutinative languages, we need a large corpus of sentences for robust parameter estimation in statistical NLP models. There have been very few efforts to build a Turkish text corpus. METU Turkish Corpus is a hand-compiled annotated collection of two million words of written Turkish samples [73]. While this corpus is useful for computational and corpus linguistics studies, it is very limited in size and coverage to be successfully used in many statistical natural language applications. Another effort is the collection of web pages of Turkish newspapers containing about 2.5 million words for Turkish speech recognition research [74]. This corpus is also limited in size and the corpus collection and development process has not been described. Although there exist some other text corpora used in Turkish language research (such as those from SABANCI University and METU), to the best of our knowledge, the compilation processes have not been documented.

In this chapter, we describe the language resources that we built for processing the Turkish morphology and give some applications that utilize these resources. We make the following language resources available for research purposes[4] :

[4] All resources are available at http://www.cmpe.boun.edu.tr/~hasim

- A stochastic finite-state morphological parser: It is a weighted lexical transducer that can be used for morphological analysis and generation of words. The transducer has been stochastized using the morphological disambiguator and the web corpus. The parser can be used with the OpenFST weighted finite-state transducer library [75].
- An averaged perceptron-based morphological disambiguator: The proposed system has the highest disambiguation accuracy reported in the literature for Turkish. It also provides great flexibility in features that can be incorporated into the disambiguation model, parameter estimation is quite simple, and it runs very efficiently. It can also be implemented as a weighted finite-state machine [34].
- A web corpus (a corpus collected from the web): We aimed at collecting a representative sample of the Turkish language as it is used on the web. This corpus is the largest web corpus for Turkish. The compilation process and corpus statistics are described in detail.

In order to observe the effectiveness of these resources in language processing, we implemented two language applications. The first one is a spell checker which uses the stochastic morphological parser (stochastized using the disambiguator and the corpus) as a computational lexicon. Since the parser also outputs the probability of a word during morphological analysis, we can order the spelling suggestions for the misspelled words. As another application, we used the morphological parser to build a morphology-based language model for broadcast news transcription.

3.1. Finite-State Morphological Parser

In morphologically rich languages, grammatical features and functions, which are associated with the syntactic structure of a sentence in other types of language, are often represented within the morphological structure of a word in addition to the syntactic structure. Therefore, in these languages, we need a morphological parser to break a word into its constituent morphemes annotated with morphosyntactic and morphosemantic features.

Table 3.1. Operator types and their explanations.

Operator type	Explanation
a:b ⇐ c _ d	a is **always** realized as b in the context c _ d
a:b ⇒ c _ d	a may be realized as b **only** in the context c _ d
a:b ⇔ c _ d	a must be realized as b in the context c _ d and nowhere else
a:b / ⇐ c _ d	a is **never** realized as b in the context c _ d

In Turkish, theoretically one can produce an infinite number of words by inserting some derivational suffixes like the causative suffix in a word multiple times. Even if we ignore such iterations which are rarely used in practice, we can generate a word like the following using each suffix only once:

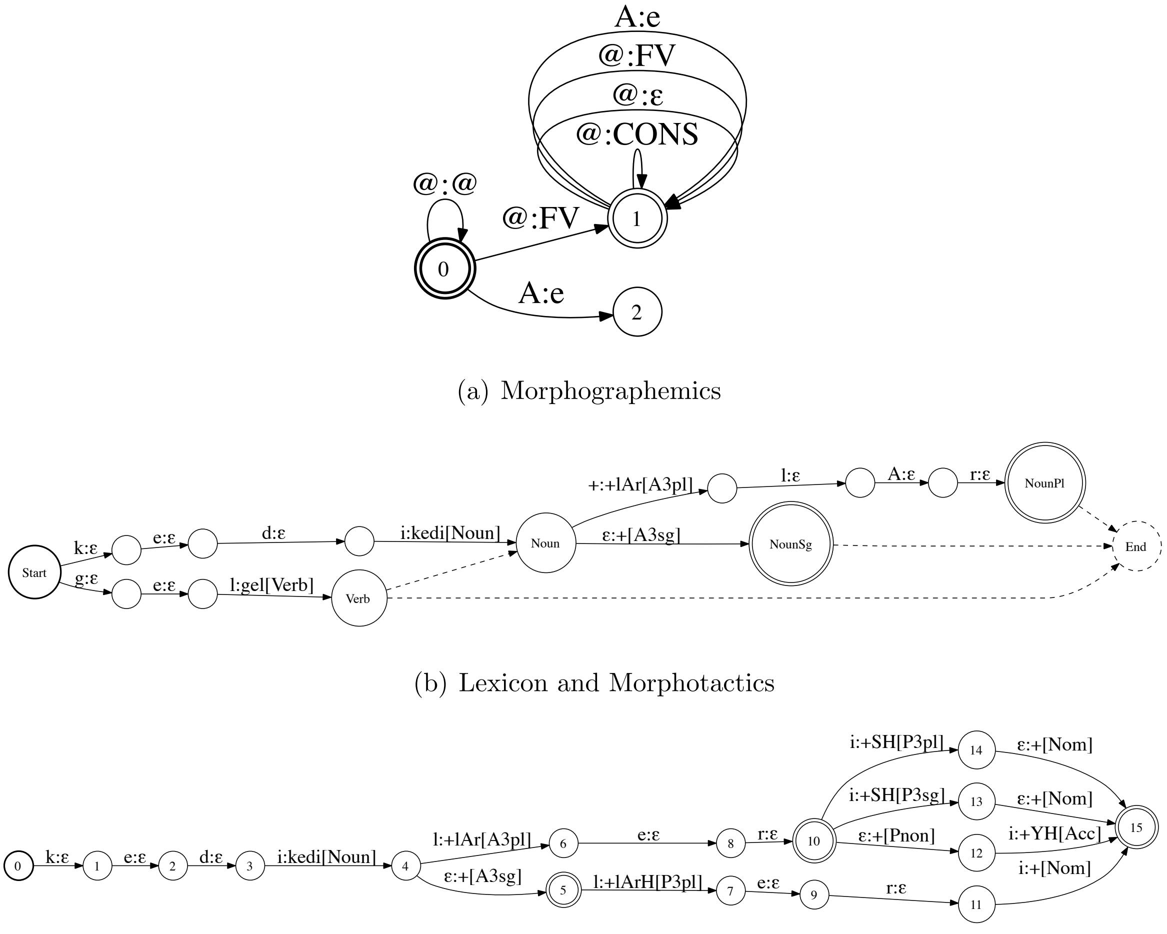
ölümsüzleştiriveremeyebileceklerimizdenmişsinizcesine
"(behaving) as if you are among those whom we could not cause hastily to become immortal"

We can break this word into morphemes as shown below:

ölümsüz+leş+tir+iver+eme+yebil+ecek+ler+imiz+den+miş+siniz+cesine

In order to build a morphological parser, we need three components: a lexicon listing the stem words annotated with some information such as the part-of-speech tags to determine which morphological rules apply to them, a morphotactics component (morphosyntax) that describes the word formation rules by specifying the ordering of morphemes, and a morphophonemics component that describes the phonological alternations occurring in the morphemes during word formation. All these components can be implemented using finite-state transducers (FSTs).

To implement the phonological rules, we used the two-level morphology formalism of Koskenniemi [76], which is a framework for describing morphological alternations. In this formalism, the phonological rules denote regular relations that can be represented by finite-state transducers. Two-level rules are applied in parallel or when implemented as finite-state transducers they can be combined into a single morphophonemics trans-



(a) Morphographemics

(b) Lexicon and Morphotactics

(c) Lexical transducer

Figure 3.1. (a) Turkish vowel harmony rule example: "@" symbol represents any absent feasible lexical or surface symbol. (b) Turkish nominal inflection example (c) Lexical transducer showing ambiguous parses for the word *kedileri*.

ducer. In two-level phonology, the phonological rules use the four operators shown in Table 3.1. The rules are declarative constraints for the lexical to surface mapping of symbols. The lexical form of morphemes represents a common underlying structure. The surface form represents the orthographic realization of the lexical forms of morphemes. In this formalism, there is one-to-one correspondence between the lexical and surface form symbols. The underscore symbol in the table indicates the position of symbol pair (a:b) between the left (c) and right (d) contexts.

The following rule is an example for vowel harmony phenomena in Turkish which forces change of vowels in surface form of suffixes to agree in backness with the preceding vowel.

```
A:e ⇒ @:FV [ :CONS | :0 ]* _
```

This rule states that symbol "A" in lexical level may be converted to /e/ vowel only if it is preceded with a surface front vowel followed possibly by a number of symbols having consonant or epsilon realizations in the surface level. Finite-state transducer implementation of this rule in a compact form can be seen in Figure 3.1a. The compilation and intersection of all the rule transducers as finite-state automata is a morphographemics transducer.

The morphotactics which encodes the morphosyntax - the ordering of morphemes - can also be represented as a finite-state machine. Figure 3.1b shows a small part of the lexicon and morphotactics for Turkish represented as a finite-state transducer. We refer to this transducer as morphotactics transducer. The finite-state transducer of the morphological parser is obtained as the composition of the morphographemics transducer and morphotactics transducer. Figure 3.1c shows the part of this lexical transducer corresponding to all the parses of the ambiguous word *kedileri*. The two-level phonological rules and the morphotactics were adapted from the PC-KIMMO implementation of Oflazer [60]. The phonological rules and the morphotactics have been expanded and modified to cover the phenomena and the exceptions not handled in the PC-KIMMO implementation. The rules were compiled using the *twolc* rule compiler [62]. We used the morphosyntactic tag set of Oflazer [77]. A new root lexicon of 54,267 words based on the Turkish Language Institution dictionary[5] was compiled.

The two-level rules [76] that describe the phonological alternations in Turkish are given in Appendix A. These phonological rules are compiled into a finite-state transducer [78]. For this purpose, we used the Xerox two-level rule compiler [62, 79]. Appendix B shows the verbal and nominal morphotactics for Turkish. We composed the lexicon/morphotactics transducer with the morphophonemics transducer which is the intersection of the phonological rule transducers to build the lexical transducer

[5] http://www.tdk.gov.tr

of the parser [80]. We used AT&T FSM tools [5] for finite-state operations. The resulting finite-state transducer can also be used with the OpenFST weighted finite-state transducer library [75].

We show below the morphological analysis of the word mentioned previously in this section as an example:

> *ölümsüz*[Adj]-*lAş*[Verb+Become]-*DHr*[Verb+Caus]+[Pos]
> -*YHver*[Verb+Hastily]+*YAmA*[Able+Neg]-*YAbil*[Verb+Able]
> -*YAcAk*[Noun+FutPart]+*lAr*[A3pl]+*HmHz*[P1pl]+*NDAn*[Abl]
> -*YmHş*[Verb+Narr]+*sHnHz*[A2pl]-*CAsHnA*[Adv+AsIf]

The morphological representation is similar to the one used in [37]. Each output of the parser begins with the root word and its part-of-speech tag in brackets. These are followed by a set of lexical morphemes associated with morphological features (nominal features such as case, person, and number agreement; verbal features such as tense, aspect, modality, and voice information). The inflectional morphemes start with a + sign. The derivational morphemes start with a - sign and the first feature of a derivational morpheme is the part-of-speech of the derived word form. A morphological feature may be appended without any morpheme, indicating that the feature is also applicable to the current word form.

The word coverage rate of the morphological parser is about 96.7% on the text corpus collected from online newspapers (see Table 3.6). The parser can also recognize the punctuation marks and the numerical tokens. It is highly efficient and can analyze about 8,700 words per second on a 2.33 GHz Intel Xeon processor. As explained in Section 3.4, the morphological parser was also converted into a stochastic parser, which makes it the first stochastic morphological parser for Turkish.

3.2. Morphological Disambiguation

The morphological parser may return more than one possible analysis for a word due to ambiguity. For example, the parser outputs four different analyses for the word *kedileri* as shown below. The English glosses are given in parentheses.

kedi[Noun]+*lAr*[A3pl]+*SH*[P3sg]+[Nom] (his/her cats)
kedi[Noun]+*lAr*[A3pl]+[Pnon]+*YH*[Acc] (the cats)
kedi[Noun]+*lAr*[A3pl]+*SH*[P3pl]+[Nom] (their cats)
kedi[Noun]+[A3sg]+*lArH*[P3pl]+[Nom] (their cat)

This morphological ambiguity needs to be resolved for further language processing. Morphological disambiguation can be considered as morphosyntactic tagging in agglutinative languages in analogy to part-of-speech tagging in Indo-European languages. In this respect, we employ a discriminative training algorithm for learning a disambiguation model.

3.2.1. Methodology

The problem of finding the most likely morphological analyses of the words in a sentence can be solved by estimating some statistics over the parts of the morphological analyses on a training set and then choosing the most likely parse output using the estimated parameters. For parameter estimation, we use the averaged perceptron algorithm. We decided on using the perceptron method since it is very flexible in features that can be incorporated in the model and the parameter estimation method is quite easy which just requires additive updates to a weight vector.

We presented an application of the averaged perceptron algorithm to morphological disambiguation of Turkish text in a previous study [72]. In that study, a baseline trigram-based model [70] is used to enumerate n-best candidates of alternative morphological parses of a sentence. Then the averaged perceptron algorithm is applied to rerank the n-best candidate list using a set of features. In the present study, we do not use a baseline model to generate n-best candidates. Instead, we do a Viterbi decoding [28] of the best path in the network of ambiguous morphological parses of the words in a sentence using the averaged perceptron algorithm to train the model parameters, as explained in the next subsection.

We split the morphological analysis of a word into morphemic units to be used as features by the perceptron algorithm. For this purpose we make use of the morpheme boundaries (both inflectional and derivational ones) in the analysis. This representation is different than the one used by [70] and [72], where only derivational boundaries are used to split the morphological analysis of a word into chunks called inflectional groups.

A morphosyntactic tag t_i, which is a morphological analysis of a word w_i, is split into a root tag r_i and a morpheme tag m_i. The morpheme tag m_i is the concatenation of the morphosyntactic tags of morphemes $m_{i,j}$ for $j = 1 \ldots n_i$, where n_i is the number of morphemes in t_i:

$$t_i = r_i m_i = r_i m_{i,1} m_{i,2} \ldots m_{i,n_i}$$

For example, the morphological analysis of the word $w_i =$*ulaşmadığı*

$t_i =$*ulaş*[Verb]+*mA*[Neg]-*DHk*[Noun+PastPart]+[A3sg]+*SH*[P3sg]+[Nom]

is represented as its root tag and morpheme tags as follows:

$r_i =$*ulaş*[Verb]
$m_{i,1} =$+*mA*[Neg]
$m_{i,2} =$-*DHk*[Noun+PastPart]+[A3sg]
$m_{i,3} =$+*SH*[P3sg]+[Nom]

The set of features that we incorporate in the model is a subset of the features used by [72]. The feature set takes into account the current morphosyntactic tag t_i, the previous tag t_{i-1}, and the two previous tag t_{i-2}. The feature templates are given in Table 3.2. We basically add unigram, bigram and trigram features over the root and the morpheme tags. The discriminative training algorithm of the perceptron learns the feature weights for each instance of these features.

Table 3.2. Feature templates used for morphological disambiguation.

Gloss	Feature
Morphological parse trigram	(1) $t_{i-2}t_{i-1}t_i$
Morphological parse bigram	(2) $t_{i-2}t_i$ & (3) $t_{i-1}t_i$
Morphological parse unigram	(4) t_i
Morpheme tag with previous tag	(5) $t_{i-1}m_i$
Morpheme tag with two previous tag	(6) $t_{i-2}m_i$
Root trigram	(7) $r_{i-2}r_{i-1}r_i$
Root bigram	(8) $r_{i-2}r_i$ & (9) $r_{i-1}r_i$
Root unigram	(10) r_i
Morpheme tag trigram	(11) $m_{i-2}m_{i-1}m_i$
Morpheme tag bigram	(12) $m_{i-2}m_i$ & (13) $m_{i-1}m_i$
Morpheme tag unigram	(14) m_i
Individual morpheme tags	(15) $m_{i,j}$ for $j = 1 \ldots n_i$
Individual morpheme tags with position	(16) $jm_{i,j}$ for $j = 1 \ldots n_i$
Number of morpheme tags	(17) n_i

3.2.2. Perceptron Algorithm

A variant of the perceptron algorithm that can be applied to problems like tagging and parsing is repeated in Figure 3.2 from Collins [1]. The algorithm estimates a parameter vector $\vec{\alpha}$ using a set of training examples (x_i, y_i), which will be used for mapping from inputs $x \in X$ to outputs $y \in Y$. In our setting, X is a set of sentences and Y is a set of possible morphological parse sequences. The algorithm makes multiple passes (denoted by T) over the training examples. For each example, it finds the highest scoring candidate among all candidates using the current parameter values. If the highest scoring candidate is not the correct parse, it updates the parameter vector $\vec{\alpha}$ by the difference of the feature vector representation of the correct candidate and the highest scoring candidate. This way of parameter update increases the parameter values for features in the correct candidate and downweights the parameter values for features in the competitor. For the application of the model to the test examples, the algorithm calculates the "averaged parameters" since they are more robust to noisy

```
**input** set of training examples $\{(x_i, y_i) : 1 \leq i \leq N\}$
**input** number of iterations $T$
$\vec{\alpha} = 0$, $\vec{\gamma} = 0$
**for** $t = 1 \ldots T$, $i = 1 \ldots N$ **do**
    $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \vec{\alpha}$
    **if** $z_i \neq y_i$ **then**
        $\vec{\alpha} = \vec{\alpha} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
    **end if**
    $\vec{\gamma} = \vec{\gamma} + \vec{\alpha}$
**end for**
**return** $\vec{\gamma} = \vec{\gamma}/(nT)$
```

Figure 3.2. The averaged perceptron algorithm by Collins [1].

or inseparable data [1]. The averaged parameters $\vec{\gamma}$ are calculated by summing the parameter values for each feature after each training example and dividing this sum by the total number of examples used to update the parameters.

The perceptron algorithm is adapted to the morphological disambiguation problem as follows:

- The training examples are the pairs (x_i, y_i) for $i = 1 \ldots n$, where n is the number of training sentences. For the i^{th} sentence, x_i is the word sequence $w^i_{[1:n_i]}$ and y_i is the correct morphosyntactic tag sequence $t^i_{[1:n_i]}$, where n_i is the number of words in the sentence.
- The function $\mathbf{GEN}(x_i)$ maps the input sentence x_i to the candidate parse sequences. We consider all possible combinations of the alternative morphological analyses of the words in the sentence. In the actual implementation, we do not enumerate all possible morphological parse sequences. Since the features depend on the current and previous two tags, we generate a network of parse outputs on-the-fly and do a Viterbi decoding of the best path without enumerating all the paths.
- The representation $\mathbf{\Phi}(x, y) \in \Re^d$ is a d-dimensional feature vector. Each compo-

nent $\Phi_j(w_{[1:n]}, t_{[1:n]})$ for $j = 1 \ldots d$ is the count of a local feature (n is the number of words in the word sequence x) and is defined as $\sum_{i=1}^{n} \phi_j(t_{i-2}, t_{i-1}, t_i)$, where $\phi_j(t_{i-2}, t_{i-1}, t_i)$ is an indicator function for the j^{th} feature. The features depend on the current morphosyntactic tag (morphological parse) and the history of the previous two tags. An example indicator function for a feature (corresponding to feature template (9) in Table 3.2) might be:

$$\phi_{100}(t_{i-2}, t_{i-1}, t_i) = \begin{cases} 1 & \text{if the root tag of } t_i \\ & \text{is gör[Verb] and} \\ & \text{the root tag of } t_{i-1} \\ & \text{is uygun[Adj]} \\ 0 & \text{otherwise} \end{cases}$$

For this example, $\Phi_{100}(w_{[1:n]}, t_{[1:n]})$ is the number of times the root tag *uygun*[Adj] is followed by the root tag *gör*[Verb] in the tag sequence $t_{[1:n]}$. Then, the algorithm will update the 100^{th} component of the parameter vector $\vec{\alpha}$ by adding $\Phi_{100}(x, y) - \Phi_{100}(x, z)$, where y is the correct tag sequence and z is the highest scoring candidate.

- The expression $\mathbf{\Phi}(x, y) \cdot \vec{\alpha}$ denotes the inner product $\sum_{j=1}^{d} \Phi_j(x, y)\alpha_j$, where α_j is the j^{th} component of the parameter vector $\vec{\alpha}$.
- The function $\arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \vec{\alpha}$ can be efficiently calculated using dynamic programming since the features that we use depend on only the current tag and the previous two tags.

With this setting, the perceptron algorithm learns an averaged parameter vector $\vec{\gamma}$ that is used to choose the most likely morphological parse sequence of a test sentence x using the following function:

$$\begin{aligned} F(x) &= \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \vec{\gamma} \\ &= \arg\max_{y \in \mathbf{GEN}(x)} \sum_{j=1}^{d} \Phi_j(x, y)\gamma_j \end{aligned}$$

Convergence theorems for the perceptron algorithm applied to tagging and parsing problems are given by Collins [1].

3.2.3. Experiments and Results

In order to measure the performance of the morphological disambiguator, we used a semi-automatically disambiguated Turkish corpus of about 950000 tokens (including markers such as begin and end of sentence markers) that has been tagged with Oflazer's parser. In addition to the correct morphological analysis of a word, alternative ambiguous analyses are also available in the corpus given as output by the morphological analyzer.

The output format of the morphological parser developed in this work is somewhat different from that used in the corpus due to the improvements on the phonological rules and the morphotactics. Therefore, we converted the corpus to the parser format used in this work in order to be able to train a disambiguation model over the outputs of our morphological parser. In this conversion, the morphological analysis of 95.5% of the tokens were mapped automatically to an analysis in the form used by the morphological parser. The 2.6% of the tokens (mostly misspelled words and proper nouns) could not be parsed and they were marked as unknown words. The remaining 1.9% of the tokens were mapped to an analysis which has the minimum edit distance with the original parse.

This data set was divided into training, development and test sets with respective sizes of about 850000 (45000), 47000 (2500) and 48000 (2,500) tokens (sentences). The training set was used for parameter estimation and the development set was used for feature selection. The trained model was tested on this semi-automatically disambiguated test set. In addition to this test set, the same trained model was also evaluated on a manually disambiguated test set (converted into out parser format) of 958 tokens.

The results of the disambiguation experiment are given in Table 3.3. 95% con-

Table 3.3. Morphological Disambiguation Results.

Test Set	Accuracy (%)
Semi-automatically tagged (48K tokens)	97.05
Manually tagged (958 tokens)	96.45

Table 3.4. Comparative Results on Manually Tagged Test Set (958 tokens).

Method	Accuracy (%)
Trigram-based model [70]	95.92
GPA [71]	95.82
Perceptron (this study)	96.45

fidence interval for the 97.05% accuracy on the semi-automatically tagged test set is [0.9689, 0.9720]. On the manually tagged test set, the confidence interval for the 96.45% accuracy is [0.9526, 0.9763].

We also evaluated the performance of the perceptron algorithm on the original corpus. The reason for this is to compare the performance of the algorithm to previous studies on this corpus. The results are given in Table 3.4. The trigram-based model of [70] is our implementation. The confidence interval for the 96.45% accuracy of the perceptron method is [0.9526, 0.9763]. The performance improvement of the perceptron model is not statistically significant on this small data set. It seems that we need a larger manually disambiguated corpus to verify the performance improvements.

The morphological disambiguator is also a part-of-speech (POS) tagger when we consider the POS tag of a word as the POS tag of the last derived word form as given in the morphological parse of the word. The POS tagging performance of the disambiguator is about 98.6% for both test sets. An example sentence from the disambiguated test corpus is given in Table 3.5.

The perceptron algorithm trains a disambiguation model by making four passes over the training examples of about one million tokens in one hour on a 2.33 GHz Intel Xeon processor. The generated model contains about 580000 features - this is

Table 3.5. An example of a morphologically disambiguated sentence. The first morphological parse for each word is the analysis the disambiguator chooses.

Word	Parses
<S>	<S>+BSTag
Ancak	ancak[Conj] ancak[Adv]
Merkez	merkez[Noun]+[A3sg]+[Pnon]+[Nom]
Bankası	banka[Noun]+[A3sg]+SH[P3sg]+[Nom]
TL	TL[Noun]+[Acro]+[A3sg]+[Pnon]+[Nom]
yönünden	yön[Noun]+[A3sg]+SH[P3sg]+NDAn[Abl] yön[Noun]+[A3sg]+Hn[P2sg]+NDAn[Abl] Yön[Noun]+[Prop]+[A3sg]+SH[P3sg]+NDAn[Abl] Yön[Noun]+[Prop]+[A3sg]+Hn[P2sg]+NDAn[Abl]
rahatlarken	rahatla[Verb]+[Pos]+Hr[Aor]+[A3sg]-Yken[Adv+While] rahat[Adj]-[Noun]+lAr[A3pl]+[Pnon]+[Nom]-[Verb]-Yken[Adv+While] rahat[Noun]+lAr[A3pl]+[Pnon]+[Nom]-[Verb]-Yken[Adv+While]
,	,[Punc]
döviz	döviz[Noun]+[A3sg]+[Pnon]+[Nom]
rezervlerinde	rezerv[Noun]+lAr[A3pl]+SH[P3sg]+NDA[Loc] rezerv[Noun]+[A3sg]+lArH[P3pl]+NDA[Loc] rezerv[Noun]+lAr[A3pl]+SH[P3pl]+NDA[Loc] rezerv[Noun]+lAr[A3pl]+Hn[P2sg]+NDA[Loc]
erime	erim[Noun]+[A3sg]+[Pnon]+YA[Dat] eri[Verb]+[Pos]-mA[Noun+Inf2]+[A3sg]+[Pnon]+[Nom] Er[Noun]+[Prop]+[A3sg]+Hm[P1sg]+NA[Dat] eri[Verb]+mA[Neg]+[Imp]+[A2sg] er[Adj]-[Noun]+[A3sg]+Hm[P1sg]+NA[Dat] er[Noun]+[A3sg]+Hm[P1sg]+NA[Dat]
gözlendi	gözle[Verb]-Hn[Verb+Pass]+[Pos]+DH[Past]+[A3sg] göz[Noun]+[A3sg]+[Pnon]+[Nom]-lAn[Verb+Acquire]+[Pos] +DH[Past]+[A3sg]
</S>	</S>+ESTag

the dimension of the feature vector. It can disambiguate 1000 words per second on the same processor.

3.3. Web Corpus

In the domain of language processing, we need large corpora for the application and evaluation of statistical methods. Such corpora are also important for empirical methods that the linguists and lexicographers use to infer information about language. For example, in statistical NLP applications, a large amount of data is necessary to reliably estimate the language model parameters. The situation is more severe in the case of morphologically complex languages, for which acceptable performance rates for the proposed models can only be attained using quite large corpora.

It has been reported that probabilistic models of language based on a very large corpus, even if the corpus is noisy, are better than those based on estimates from smaller, cleaner data sets [81]. The web is being increasingly used by researchers to build web corpora [82]. Some of the reasons of this tendency are that the other language resources are not large enough or not suitable in terms of language coverage and the web is instantly and freely available.

In this research, we built a large corpus for Turkish and cleaned it using the morphological parser and some heuristics. The corpus is composed of four subcorpora. Three of these corpora (referred as *NewsCor*) are from three major news portals in Turkish. The other corpus (referred as *GenCor*) is a general sampling of Turkish web pages. The combined corpus of these subcorpora will be referred as *BOUN Corpus*. The web corpus is intended to be used by computational linguists, lexicographers, and researchers working on statistical methods in language processing. As the texts on the web contain a large number of proper nouns, we also plan to use the web corpus to increase the lexicon size of the morphological parser. In the rest of this section, we first explain the compilation process and then we give statistics about the contents of the corpus.

3.3.1. Web Crawling

For data collection from the web, we implemented a web crawler - an automated script to browse the web as used by the search engines. The web crawler starts with a set of seed URLs. As these URLs are visited, the new hyperlinks in the downloaded pages are extracted and added to the list of URLs to visit. For *NewsCor*, seed URLs were generated automatically according to the news portal's page naming policy. We also ensured that the web spider stays in the same domain by filtering the newly discovered URLs. For *GenCor*, we started with the seed URLs in the Google directory index for Turkish. There exist about 2,900 URLs in this directory. For this corpus, we limited the number of URLs from a domain to 10000 in order to gather a general sample of the web and we downloaded about 8000000 web pages.

The general algorithm for the web crawler is as follows. We store the URLs to be visited in a database table. We fetch a random URL from this table and download the corresponding web page. The visited URLs are stored in another table in order to prevent downloading the same page again. To collect only Turkish pages, we require that either the page encoding is Turkish or if it is unicode encoded the domain name has Turkey country code. Since this does not ensure that the contents will be in Turkish, later in the text cleaning stage, we use the morphological parser to detect non-Turkish pages and we delete these pages. The text in the web pages is converted to UTF-8 encoding if it is in a different format. The HTML tags related to font rendering are removed from the text and the contents between the remaining HTML tags are extracted as separate lines of text. We also extract the hyperlinks in the page and add those that have not been visited or planned to be visited yet to the URL table. The output of the web crawler is the HTML tag-free text encoded in UTF-8.

3.3.2. Text Cleaning

Since the web corpus is very noisy, we need to do some automatic normalization and filtering to clean the corpus. The web pages contain a large amount of text not directly related to the main content, such as the text in navigation menus and advertisements. It is quite difficult for an automated agent to correctly decide the quality of the content in web pages or to decide which part of the document should be included in the corpus. We followed a multi step process to clean the corpus as described below:

(i) Decode HTML entities. For instance, the less-than sign &lt; is replaced by <.
(ii) Trim white spaces at the start and end of the lines.
(iii) Estimate statistics over letter sequences from a Turkish text and use these statistics to filter non-Turkish documents. This step removes documents that have a distribution of letter combinations which is significantly different from the distribution in Turkish.
(iv) Remove duplicate lines to get rid of repetitions in web pages, such as text in navigation menus.
(v) Remove documents with less than 1000 characters. These documents mostly contain hypertext links without any useful content.
(vi) Remove documents for which more than 25% (an empirically determined threshold) of the words cannot be parsed using the morphological parser.

The last step above is necessary since, although only pages with Turkish encoding are taken into account by the web crawler, it is not uncommon to find pages containing a significant amount of material in other languages. For instance, an advertisement in both Turkish and English is a typical case. Such pages should clearly be eliminated since they include a large amount of noise. The normalization and filtering steps removed about 60% of the text collected for *NewsCor* and 90% of the text collected for *GenCor*. This difference is expected since the web corpus data are much more noisy when compared to the news portal data.

3.3.3. Tokenization and Segmentation

The tokenization and segmentation of a corpus is often necessary in language applications. Since the corpus compiled in this work is very large for manual operation, we employed automatic methods to tokenize and segment the corpus into sentences.

We used the morphological parser developed in this study as a computational lexicon to look up the words in the corpus. If the parser can return an analysis for a text entity, we accept that entity as a token. Otherwise, we try to segment the text entity into its constituent words and symbols. For example, a word at the end of a sentence joined with a period is split into the word and period tokens. The parser can recognize abbreviations, acronyms, and numbers as tokens. We also treat some entities like URLs, e-mail addresses, dates, punctuation symbols, and quotation marks as tokens. The tokens are then segmented into sentences using a simple sentence segmentation algorithm.

3.3.4. XML Encoding

For the encoding of the web corpus, we used the XML Corpus Encoding Standard, XCES (http://xces.org) as used by [73]. The corpus was encoded in paragraph and sentence levels. We also plan to annotate the corpus linguistically in morphosyntactic level. An example paragraph from the XCES-encoded corpus is given below.

```
<p><s><q>Cthulhunun Çağrısı</q> ve ardından
<q>Deliliğin Dağlarında</q> adlı eserleri Türkçeye
çevrilen Howard Phillips Lovecraft korku ve fantezi
ustası bir yazar .</s></p>
```

Howard Phillips Lovecraft whose works "The Call of Cthulhu" and "At the Mountains of Madness" were translated into Turkish is an author skilled in horror and fantasy.

The quotation symbols in the text are used for tagging quotations. The sentences are tagged according to the output of the sentence segmentation algorithm. The paragraph tags are used for text fragments separated by the end of line character.

3.3.5. Contents of the Corpus

As stated before, Turkish web corpus is formed of four subcorpora. Three of these (Milliyet, Ntvmsnbc, Radikal) are from three major news portals in Turkish

Table 3.6. Web Corpus Size and Results of Morphological Parser.

Corpus	Words	Tokens	Types	Tokens Parsed(%)	Types Parsed(%)
Milliyet	59M	68M	1.1M	96.7	63.5
Ntvmsnbc	75M	86M	1.2M	96.4	55.8
Radikal	50M	58M	1.0M	97.0	65.7
NewsCor	184M	212M	2.2M	96.7	52.2
GenCor	239M	279M	3.0M	94.6	39.5
BOUN Corpus	423M	491M	4.1M	95.5	38.4

(collectively referred as *NewsCor*) and the other one (*GenCor*) is a general sampling of Turkish web pages. The statistics about the numbers of words (all words in the corpus), tokens (words and lexical units such as punctuation marks), and types (distinct tokens) are shown in Table 3.6. The percentages of the tokens and types in the corpus that can be successfully parsed by the morphological parser are also indicated. We can interpret the figures on the table from different points of view. First, we observe that, due to the agglutinative nature of the language, the number of types (4.1M) is quite large. Also, the number of types parsed (1.57M) in the corpus being about 30 times larger than the size of the root lexicon of the parser indicates that derived words are used commonly in Turkish.

Second, a significant difference exists between the percentages of the tokens and the types successfully parsed. This is an expected result, since most of the tokens in the corpus are grammatical words and there is a relatively small amount of other kinds of tokens (punctuation symbols, proper nouns, etc.) that cannot be parsed. On the other hand, each distinct token is treated equally in the last column of the table, without taking frequencies into consideration. We see that the parser can return an analysis only for 38.4% of the types; the rest cannot be parsed. However, this percentage of types in fact constitutes 95.5% of the corpus. The main reasons for the unparsed types are the proper nouns that do not exist in the lexicon and the spelling errors in the corpus.

Another observation is about the cleanness of the corpus. When we compare

GenCor with *NewsCor*, we notice a decrease in the percentage of words that can be parsed. The difference is about 2% in the case of tokens while it is much higher (12.7%) in the case of types. These figures indicate that *NewsCor* is much cleaner than *GenCor*, as might be expected. In addition, the analysis of the number of words, tokens, and types in the two subcorpora shows that *GenCor* includes more types that are not actually words and there are also some unparsed tokens with high frequencies on this subcorpus. These observations signal that the words used by general web users are more diverse than those used in news portals and some of these words seem to be accepted (due to their high frequencies) by the web community.

Finally, the performance ratios for the morphological parser are quite satisfactory. The success is 96.7% on *NewsCor* and it is slightly lower for *GenCor* due to the special characteristics of the written text on the web.

3.3.6. Corpus Statistics

In this section, we present statistical results about the corpus in order to get an idea about the coverage of a corpus of this size for an agglutinative language and also to observe the morphological characteristics of the Turkish language. Figure 3.3 shows statistics about the types relative to the corpus size (number of tokens) excluding the numerical tokens. As can be seen, the number of types is increasing continuously for both corpora and for the combined corpus. It seems that if the corpus size is increased beyond the current size of 491M tokens, new types will still continue to emerge. This is supported by the evidence that when the corpus size was increased from 490M to 491M, 5539 new types (of which 1009 can be parsed successfully) have been added to the corpus. This is partly due to the productive morphological structure of Turkish and partly to the rich web environment. These facts indicate that the size of the current corpus does not cover all language usage. It should be extended until at least the number of types that can be parsed becomes stable, corresponding to the situation that nearly all possible derived forms are represented in the corpus. Adding more data beyond this limit will just cause an increase in the number of special tokens (e.g. proper nouns) and misspelled words.

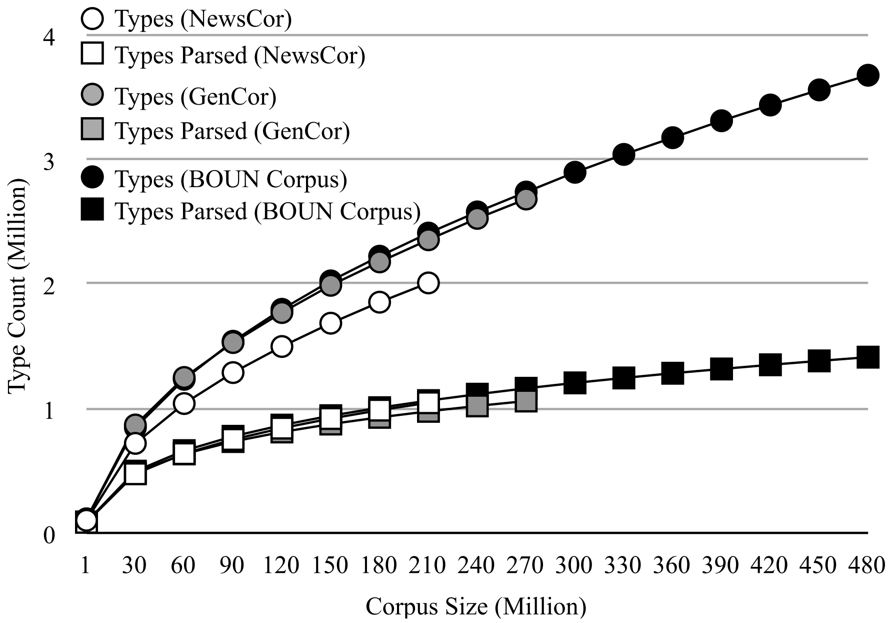


Figure 3.3. Type statistics for subcorpora and combined corpus.

Figure 3.4 shows coverage statistics with respect to the vocabulary size (number of types). The figure was obtained by first sorting the types in decreasing order of their frequencies and then summing up the frequencies beginning from the topmost entry for the indicated vocabulary sizes. 50% of the corpus is formed of only about 1000 distinct words. We observe that about 300K types are necessary in order to attain an acceptable coverage ratio (97-98%). The agglutinative nature of the language and the diversity of the web contents are the basic reasons of this result. A similar statistic related to the percentages of infrequent types shows that almost half of the types (about 2.0M) occur only once in the corpus. The number of types occurring less than 10 times is 3.4M and they represent 7.5M tokens in the corpus. Thus, we see that the majority of the types in the corpus are very infrequent.

To understand the source of the large number of types in the corpus, we give statistics for the stems and lexical endings (tokens stripped of their stems in lexical form such as *+lAr+Hn*) of the tokens that can be parsed in Figure 3.5. As the number of tokens considered reaches to the size of the corpus, the number of unique stems approaches to the size of our lexicon (55,278 root words). On the other hand, the number of unique endings increases steadily as new data are added. Note that the

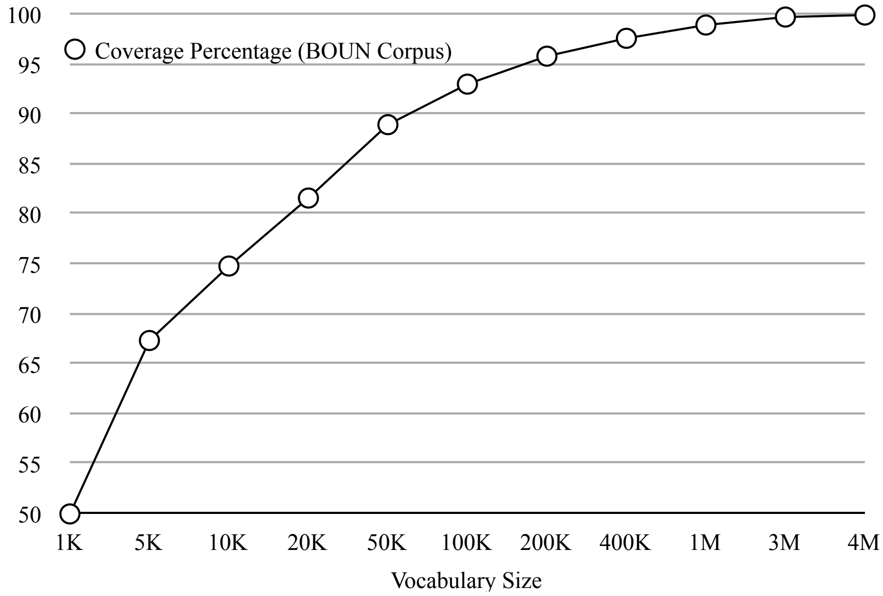


Figure 3.4. Coverage statistics for most frequent types.

figure considers only the tokens that can be successfully parsed. Hence, this increase in the number of endings means that people freely derive new word forms by making use of suffix combinations not used before. This is an interesting result. Although we know that theoretically there is no limit on the number of derivations in Turkish, we might expect that in practice a (large) subset of all possible derived forms will cover the daily use of the language. However, this expectation does not hold even for a corpus of nearly 500M tokens, and about 40 stems and 60 lexical endings emerge per 10M tokens at this size.

Figure 3.6 gives statistics about the frequency of the types that cannot be parsed. We see that 62% of the types cannot be parsed. However, the majority of the types that cannot be parsed are very infrequent. For instance, when we ignore the types that occur only once in the corpus, this ratio decreases to 26%. When we remove the types occurring less than 10 times, we can parse 95% of the types in the corpus. The types labeled as ungrammatical by the morphological parser indeed mostly result from the spelling errors in the corpus. There also exist many proper nouns that are very infrequent in the corpus.

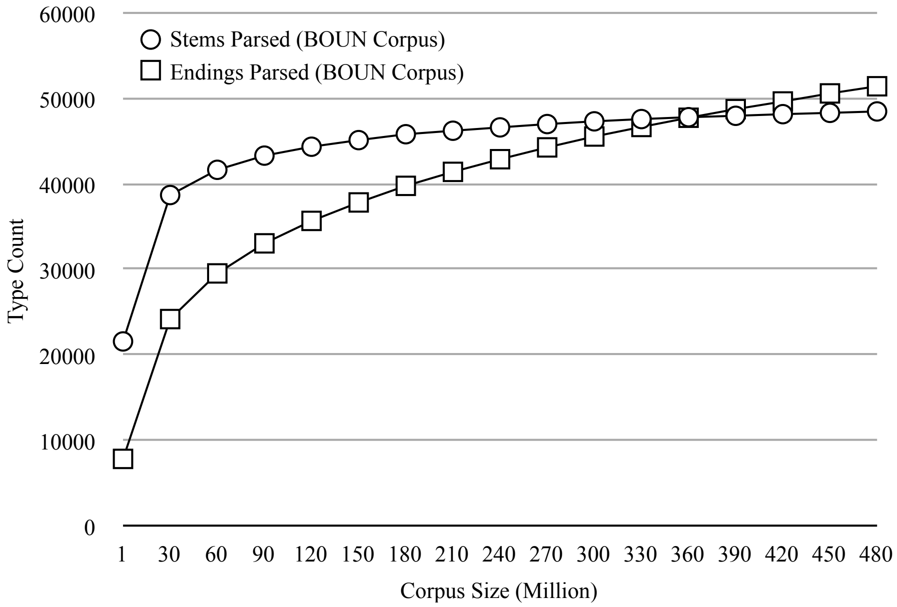


Figure 3.5. Stem and lexical ending statistics for combined corpus.

3.4. Stochastic Morphological Parser

Turkish morphology is inherently ambiguous. The average number of morphological analyses for a token in *BOUN Corpus* is 2.5 and about 65% of the tokens possess more than one analysis. In language processing applications, we often need to estimate a probability distribution over all possible analyses of words. For instance, in spell checking we can use the probability estimates of unigrams to rank misspelling suggestions for a word. In this study, we converted the morphological parser that we built into a probabilistic one in order to make it usable in such language applications.

The finite-state transducer of the morphological parser is obtained as the composition of the morphophonemics transducer *mp* and the morphotactics transducer *mt*: $mp \circ mt$. If we can estimate a statistical morphotactics model, we can convert the morphological parser to a probabilistic one. Eisner [83] gives a general EM algorithm for parameter estimation in probabilistic finite-state transducers. However, having a morphological disambiguator makes the parameter estimation easier. Since we can disambiguate the possible morphosyntactic tag sequences of a word, there is a single path in the morphotactics transducer that matches the chosen morphosyntactic tag sequence. Then the maximum-likelihood estimates of the weights of the arcs in the

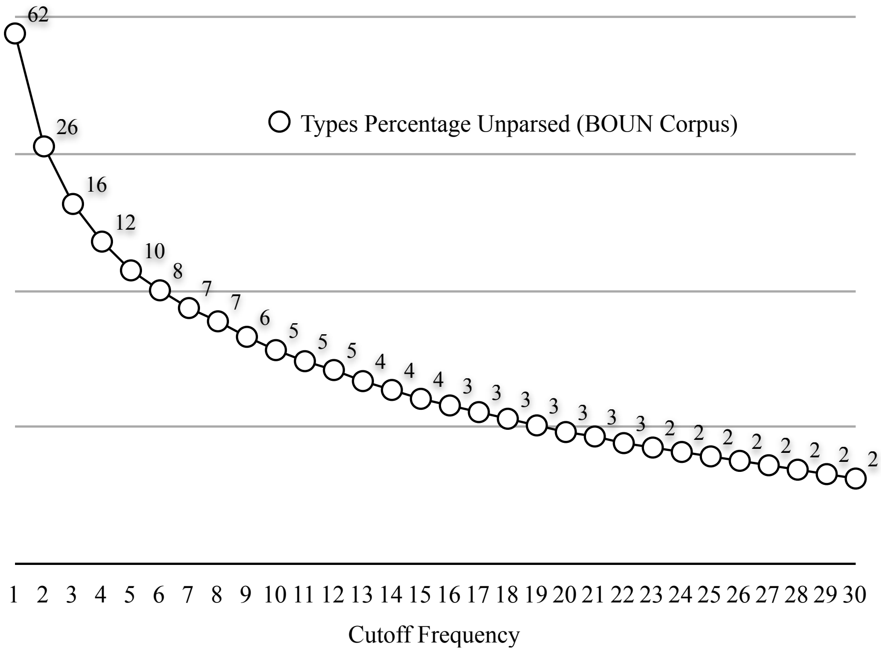


Figure 3.6. Percentages for types not recognized by the parser versus cutoff frequency.

morphotactics transducer are found by setting the weights proportional to the number of traversals of each arc and as a result making the finite-state transducer Markovian. We use a specialized semiring to cleanly and efficiently count the number of traversals of each arc.

Weights in finite-state transducers are elements of a semiring, which defines two binary operations $\otimes$ and $\oplus$, where $\otimes$ is used to combine the weights of arcs on a path into a path weight and $\oplus$ is used to combine the weights of alternative paths [11]. We define a counting semiring to keep track of the number of traversals of each arc. The weights in the mt transducer are converted to the counting semiring. In this semiring, the weights are vectors of integers and the dimension of a vector is the total number of arcs in the mt transducer. We number the arcs in the mt transducer and set the weight of the n^{th} arc as the n^{th} basis vector. The binary plus $\oplus$ and the times $\otimes$ operations of the counting semiring are defined as the sum of the weight vectors. Thus, the n^{th} value of the vector in the counting semiring just counts the appearances of the n^{th} arc of mt in a path.

To estimate the weights of the stochastic model of the mt transducer, we use the

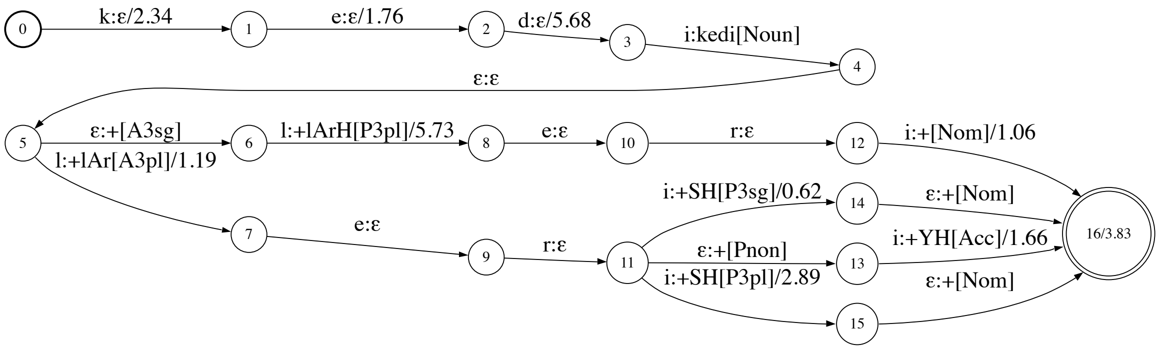


Figure 3.7. Finite-state transducer for the word *kedileri*.

text corpus collected from the web. First we parse the words in the corpus to get all the possible analyses of the words. Then we disambiguate the morphological analyses of the words to select one of the morphosyntactic tag sequences x_i for each word. We build a finite-state transducer $\epsilon \times x_i$ that maps the ϵ symbol to x_i in the counting semiring. The weights of this transducer are zero vectors having the same dimension as the mt transducer. Then the finite-state transducer $(\epsilon \times x_i) \circ (mt \times \epsilon)$ having all $\epsilon : \epsilon$ arcs is minimized to get a one-state FST which has the weight vector that keeps the number of traversals of each arc in mt. The weight vector is accumulated for all the x_i morphosyntactic tag sequences in the corpus. The final accumulated weight vector is used to assign probabilities to each arc proportional to the traversal count of the arc. We use add-one smoothing to prevent the arcs from having zero probability. The resulting stochastic morphotactics transducer $\tilde{mt}$ is composed with the morphophonemics transducer mp to get a stochastic morphological parser.

The stochastic parser now returns probabilities with the possible analyses of a word. Figure 3.7 shows the weighted paths for the four possible analyses of the word *kedileri* as represented in the stochastic parser. The weights are negative log probabilities.

3.4.1. Turkish Spell Checker

There have been some previous studies for spelling checking [84] and spelling correction [85] for Turkish text. However there has been no study to address the problem of ranking spelling suggestions. One can obviously use a stochastic morphological parser as a computational lexicon to do spelling correction and rank spelling suggestions by their probability estimates. We assume that a word is misspelled if the parser fails to return an analysis of the word. Our method for spelling correction is to enumerate all the valid and invalid candidates that resemble the incorrect input word and filter out the invalid ones with the morphological parser.

To enumerate the alternative spellings for a misspelled word, we generate all the words in one-character edit distance with the input word, where we consider insertion, deletion or substitution of one symbol, or transposition of two adjacent symbols. Although we limited the edit distance to 1, it is straightforward to allow longer edit distances. The Turkish alphabet includes six special letters (*ç*, *ğ*, *ı*, *ö*, *ş*, *ü*) that do not exist in English. These characters may not be supported in some keyboards and message transfer protocols; thus people frequently use their nearest ASCII equivalents (*c*, *g*, *i*, *o*, *s*, *u*, respectively) instead of the correct forms, such as the spelling of *nasılsın* as *nasilsin*. Therefore, in addition to enumerating words in one edit distance, we enumerate all the words from which the misspelled word can be obtained by substituting these special Turkish characters for their ASCII counterparts. For instance, for the word *nasilsin*, the alternative spellings *nasılsin*, *nasilsın*, and *nasılsın* will also be generated.

We build a finite-state transducer to enumerate and represent efficiently all the valid and invalid word forms that can be obtained by these edit operations on a word. For example, the deletion of a character can be represented by the regular expression $\Sigma^*(\Sigma!\ :!\epsilon)\Sigma^*$ which can be compiled as a finite-state transducer, where Σ is the alphabet. The union of the transducers encoding one-edit distance operations and the restoration of the special Turkish characters is precompiled and optimized with determinization and minimization algorithms for efficiency. A misspelled input word transducer is composed with the resulting transducer and in turn with the morphological parser to filter out the invalid word forms. The words with their estimated probabilities are read from the output transducer and constitute the list of spelling suggestions for the word. The probabilities are used to rank the list to display to the user. We also handle the spelling errors where omission of a space character causes joining of two correct words by splitting the word into all combinations of two substrings and checking if the substrings are valid word forms.

An example list of suggestions with the assigned negative log probabilities and their English glosses for the misspelled word *nasilsin* is given below.

nasılsın (How are you): 14.2

nakilsin (You are a transfer): 15.3

nesilsin (You are a generation): 21.0

nasipsin (You are a share): 21.2

basilsin (You are a bacillus): 23.9

On a manually chosen test set containing 225 correct words which have relatively more complex morphology and 43 incorrect words which are common misspellings, the precision and recall scores for the detection of spelling errors were measured as 0.81 and 0.93, respectively.

3.4.2. Morphology-based Unigram Language Model

The closure of the transducer for the stochastic parser can be considered as a morphology-based unigram language model. Different from standard unigram word language models, this morphology-based model can also assign probabilities to words not seen in the training corpus. It can also achieve lower out-of-vocabulary (OOV) rates than models that use a static vocabulary by employing a relatively smaller number of root words in the lexicon.

We compared the performances of the morphology-based unigram language model and the unigram word language model on a broadcast news transcription task. Note that using unigram language models in automatic speech recognition systems is not optimal. We set up this experiment to measure the effectiveness of the probability estimates of the stochastic parser in a real application. The acoustic model uses Hidden Markov Models (HMMs) trained on 183.8 hours of broadcast news speech data. The

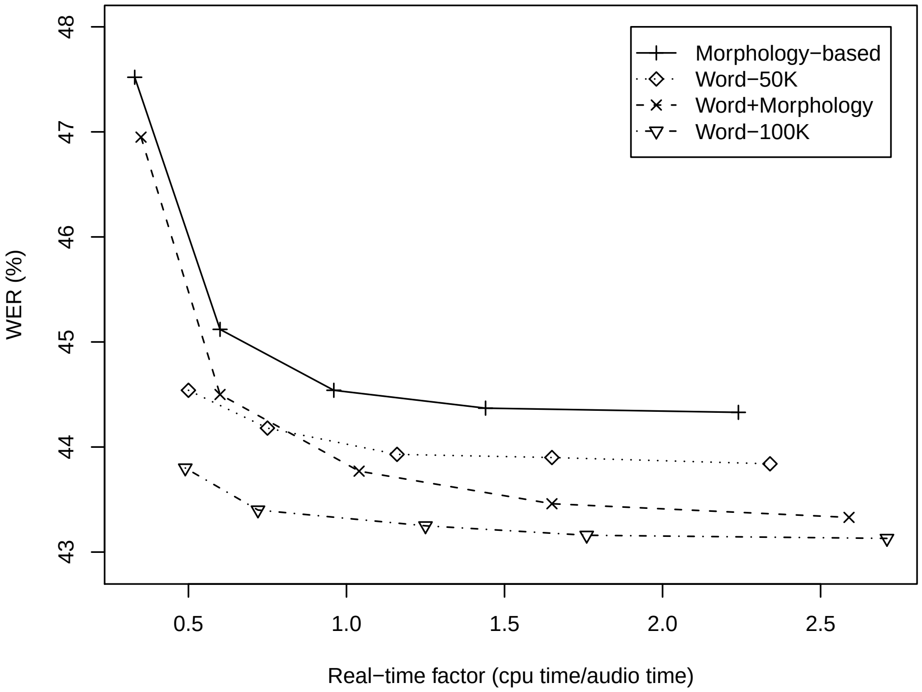


Figure 3.8. Word error rate versus real-time factor for various language models.

test set contains 3.1 hours of speech data (2410 utterances). A text corpus of 1.2 million words from the transcriptions of the news recordings was used to train the stochastic parser as explained in Section 3.4 and unigram word language models using the SRILM toolkit [39]. The automatic speech recognition system that we used for this experiment is described extensively in [38].

We experimented with four different language models. Figure 3.8 shows the word error rate versus the real-time factor for these models. In this figure, Word-50K and Word-100K are unigram word models with the specified vocabulary sizes and have the OOV rates 7% and 4.7% on the test set, respectively. The morphology-based model is based on the stochastic parser and has the OOV rate 2.8%. The 'word+morphology' model is the morphology-based model with the unigram word model probability estimates for the words seen in the training corpus and the probability estimates from the stochastic morphological parser for the unseen words. Note that the morphology-based models output the morphosyntactic tags as the recognition output which can be easily converted to words using the parser as a word generator.

Even though the morphology-based model has a better OOV rate than the word models, the word error rate (WER) is higher. One of the reasons is that the transducer for the morphological parser is ambiguous and cannot be optimized for recognition in contrast to the word models. Another reason is that the probability estimates of this model are not as good as the word models since probability mass is distributed among ambiguous parses of a word and over the paths in the transducer. The 'word+morphology' model seems to alleviate most of the shortcomings of the morphology model. It performs better than the 50K word model and is very close to the 100K word model. The main advantage of morphology-based models is that we have at hand the morphological analyses of the words during recognition.

3.5. Discussion

In this chapter, we presented some essential tools and resources for exploiting the Turkish morphology in natural language processing applications. Morphology is a very important knowledge source for morphologically complex languages like Turkish. Using these resources and tools, one can parse a text corpus and obtain the morphological analyses of the words as well as their probabilities, disambiguate the parse outputs, train statistical models using the web corpus, and build applications that fully exploit the information hidden in the morphological structure of words. As a motivation for such language applications and in order to test the effectiveness of these resources, we built a Turkish spell checker and also gave some preliminary results for morphology-based language modeling in speech recognition.

4. MORPHOLEXICAL AND DISCRIMINATIVE LANGUAGE MODELING

Language modeling for morphologically rich languages such as Arabic, Czech, Finnish, and Turkish has proven to be challenging. The out-of-vocabulary (OOV) rate for a fixed vocabulary size is significantly higher in these languages due to large number of words in language vocabulary. Having a large number of words contributes also to high perplexity numbers for standard n-gram language models due to data sparseness. These problems are especially pronounced for Turkish, being an agglutinative language with a highly productive inflectional and derivational morphology.

We can reduce the OOV rate by increasing the vocabulary size if it is not limited by the size of the text corpus available for ASR systems. However, this also increases the computational and memory requirements of the system. Besides, it may not lead to significant performance improvement due to data sparseness problem of insufficient data for robust estimation of language model parameters. Therefore, to overcome the high growth rate of vocabulary and the OOV problem, using grammatical or statistical sub-lexical units for language modeling has been a common approach. The grammatical sub-lexical units can be morphological units such as morphemes or some grouping of them such as stem and ending (grouping of suffixes). The statistical sub-lexical units can be obtained by splitting words using statistical methods. Morpheme-based language models have been proposed for German [40], Czech [41], Korean [42]. A statistical language model based on morphological decomposition of words into roots and inflectional groups which contain the inflectional features for each derived form has been proposed for morphological disambiguation of Turkish text [50]. Morphology-based language modeling approaches specifically *Factored Language Models (FLMs)* have been shown to reduce language model perplexity and lead to WER reductions in Arabic speech recognition systems [56]. FLMs decompose words into a set of features (or factors) and estimate a language model over these factors, smoothed with *generalized parallel backoff* mechanism which improves the robustness of probability estimates for rarely observed n-grams. We previously experimented with FLMs for Turkish [86] and observed that FLMs are effective in reducing perplexity of language models but only when the training data is limited. The computational cost and the inability to be represented efficiently and compactly as finite-state models also limit their usefulness. Stems and endings have been used for language modeling for Turkish [38, 54, 87] and Slovenian [44]. Statistical sub-lexical units so-called morphs have been used for language modeling of Finnish [48] and Turkish [38]. Sub-lexical language models alleviate the OOV problem, however the speech decoder can generate ungrammatical sub-word sequences and post-processing of the sub-word lattices may be required to correct the errors and increase the accuracy [54], [55]. Morphological information can also be employed later in the system as in [58], where a maximum entropy model has been trained with morphological and lexical features to rescore n-best hypotheses for Arabic speech recognition and machine translation.

We present a morphology oriented linguistic approach for language modeling in morphologically rich languages as an alternative to word and sub-word based models. This is motivated by the fact that in such languages, grammatical features and functions associated with the syntactic structure of a sentence in morphologically poor languages are often represented in the morphological structure of a word in addition to the syntactic structure. Therefore, morphological parsing of a word may reveal valuable information in its constituent morphemes annotated with morphosyntactic and morphosemantic features to exploit for language modeling.

Standard n-gram language models are difficult to beat if there is enough data. They also lead to efficient dynamic programming algorithms for decoding due to local statistics, and they can be efficiently represented as deterministic weighted finite-state automata [4]. First, we propose a novel approach for language modeling of morphologically rich languages. The proposed model, called the *morpholexical language model*, can be considered as a linguistic sub-lexical n-gram model in contrast to statistical sub-word models.

Table 4.1. Statistical and grammatical word splitting approaches.

Gloss	hello you are getting the news from the agency
word	merhaba haberleri ajanstan alıyorsunuz
morph	merhaba haber +ler +i ajans +tan al +ıyor +sun +uz
morpheme	merhaba[Noun]+[A3sg]+[Pnon]+[Nom] haber[Noun] +lAr[A3pl] +SH[P3sg]+[Nom] ajans[Noun]+[A3sg]+[Pnon] +DAn[Abl] al[Verb]+[Pos] +Hyor[Prog1] +sHnHz[A2pl]
lexical stem+ending	merhaba[Noun]+[A3sg]+[Pnon]+[Nom] haber[Noun] +lAr[A3pl]+SH[P3sg]+[Nom] ajans[Noun]+[A3sg]+[Pnon] +DAn[Abl] al[Verb]+[Pos] +Hyor[Prog1]+sHnHz[A2pl]
surface stem+ending	merhaba haber +leri ajans +tan al +ıyorsunuz

Second, we propose a novel approach to build a *morphology-integrated search network* for ASR with unlimited vocabulary in the weighted finite-state transducer framework (WFST). The proposed *morpholexical search network* is obtained by the composition of the lexical transducer of the morphological parser and the transducer of morpholexical language model. This model has the advantage of having a dynamic vocabulary in contrast to word models and it only generates valid word forms in contrast to sub-word models. The proposed model improves ASR word error rate by 1.8% absolute over word models and 0.8% absolute over statistical sub-word models at ~ 1.5 real-time factor.

And finally, we further improve ASR performance by using unigram morpholexical features in a discriminative n-best hypotheses reranking framework with a variant of the perceptron algorithm. The perceptron algorithm is tailored for reranking recognition hypotheses by introducing error rate dependent loss function. The improvements of the first-pass in WER are preserved in the reranking as 2.2% absolute over word models and 0.7% absolute over statistical sub-word models.

4.1. Generative Language Models

In the following sections, we describe the word, sub-word and morpholexical language models. The corresponding statistical and grammatical splitting approaches are shown for an example sentence in Table 4.1.

4.1.1. Word and Statistical Sub-word Language Models

The conventional approach for language modeling is estimating a statistical n-gram language model over a fixed vocabulary of words. As a baseline word language model, we built 200K vocabulary 3-gram language model which is also used as a baseline in [38].

For unlimited vocabulary speech recognition, splitting words into morpheme-like sub-words, morphs, using an unsupervised algorithm based on Minimum Description Length principle has been very effective by alleviating OOV problem and reducing language model perplexity [48]. The baseline algorithm introduced in [46] is used to segment word types in the text corpus. We used the best performing segmentations of the study in [38]. Statistical morphs have the advantage that no linguistic knowledge is required about the language. On the other hand, since morphs do not generally correspond to grammatical morphemes, we can not easily employ linguistic information in later stages of processing such as rescoring sub-word lattices. Moreover, speech decoder can generate ungrammatical sub-word sequences and post-processing of the sub-word lattices are required to correct the errors and increase the accuracy [54], [55].

4.1.2. Morpholexical Language Models

In this section, we introduce a linguistic approach to exploit morphology and alleviate OOV problem in language modeling. This can be considered as a grammatical sub-lexical language modeling approach. The modeling units are lexical and grammatical morphemes annotated with morphosyntactic and morphosemantic features. This is motivated by the fact that lexical and grammatical morphemes (*morpholexical* *units*) constitute natural sub-lexical units of a morphologically complex language. For instance, the constituent morphemes are generally the output symbols of a morphological parser when represented as finite-state models.

The morpholexical language modeling can be considered as replacing a static lexicon of words or sub-words with a dynamic computational lexicon. The dynamic lexicon over grammatical and lexical morphemes greatly solves the OOV problem by providing a root lexicon with a good coverage and makes it unnecessary to list all word forms that can be generated from a root word, which may not be even possible for languages like Turkish. For instance, the OOV rate of the morphological parser is 1.3% on the test set. This model also provides better probability estimates for rarely seen or unseen word n-grams by morphological decomposition of words.

We can train the morpholexical language models as standard n-gram language models over morpholexical units. For this, we need to parse a text corpus to get the morpholexical units using a morphological parser. Since the morphological parser can give multiple analyses due to morphological ambiguity, we need to use a morphological disambiguator to choose the correct parse of the words using the contextual information. We can then split the morphological analyses of words at morpheme boundaries and use standard n-gram estimation methods to train a language model over morpholexical units.

We also experimented with combining grammatical morphemes to build a lexical stem+ending model to alleviate the problem of large number of morphemes preventing n-grams to have a proper coverage of context. Using lexical units rather than surface forms as in statistical morphs is also beneficial in terms of decreasing data sparsity since a lexical morpheme may be realized in multiple surface forms due to phonological alternations [87]. Such an example for Turkish is the lexical plural morpheme *+lAr* which can have the surface form of *ler* or *lar* depending on the previous vowel this morpheme is suffixed. In this study, we use the lexical stem+ending decompositions to obtain the surface form stem+ending decompositions of words which can be considered as grammatical sub-words in contrast to statistical sub-words. The different modeling units for morpholexical language models can be seen in Table 4.1.

The morpholexical language models have the advantage that when combined with the lexical transducer of the morphological parser, they give probability estimates for only valid word sequences. This is not possible with statistical sub-word model or surface form stem+ending model. But, this is possible with morpholexical language models since the morphotactics effectively constrains the language model over valid morpheme sequences. We show the effect of morphotactics in language modeling by giving experimental results where we relax the morphotactics to allow any morpheme sequences in the lexical transducer. We also study the effect of morphological disambiguation in language modeling by comparing the proper morphological disambiguation of training corpus and choosing the morphological parse with the least number of morphemes.

4.2. Morpholexical Search Network for ASR

In this section, we explain how a morpholexical language model can be integrated into speech recognition in the finite-state transducer framework.

The weighted finite-state transducers (WFSTs) provide a unified framework for representing different knowledge sources in ASR systems [6]. In this framework, the speech recognition problem is treated as a transduction from input speech signal to a word sequence. A typical set of knowledge sources consists of a hidden Markov model H mapping HMM state ID sequences to context-dependent phones, a context-dependency network C transducing context-dependent phones to context-independent phones, a lexicon L mapping context-independent phone sequences to words, and a language model G assigning probabilities to word sequences. The composition of these models $H \circ C \circ L \circ G$ results in an all-in-one search network that directly maps HMM state ID sequences to weighted word sequences.

The morphology as another knowledge source can be represented as a WFST and can be integrated into the WFST framework of an ASR system. The lexical transducer of the morphological parser maps the letter sequences to lexical and grammatical morphemes annotated with morphological features. The lexical transducer can be considered as a computational dynamic lexicon in ASR in contrast to a static lexicon. The computational lexicon has some advantages over a fixed-size word lexicon. It can generate many more words using a relatively smaller number of root words in its lexicon. So it achieves lower OOV rates. Different than the static lexicon, even if we have never seen a specific word in the training corpus, the speech decoder has the chance to recognize that word. Another benefit of the computational lexicon is that it outputs the morphological analysis of the word generated. We can exploit this morphological information in a language model.

Since most of the words in Turkish have almost one-to-one mapping between graphemics and pronunciation, we use the Turkish letters as our phone set in Turkish ASR [6] . In the WFST framework, the lexical transducer of the morphological parser can be considered as a computational lexicon M replacing the static lexicon L. The transducer M outputs some symbols representing morphological features not corresponding to any lexical form in addition to lexical and grammatical morphemes. The morpholexical language model is estimated over some combination of these features and morphemes. Therefore, we need an intermediate transducer T to do the symbol mapping between these models. Then the search network with the morpholexical language G_{mlex} model can be built as $H \circ C \circ M \circ T \circ G_{mlex}$.

The WFST offers finite-state operations such as *composition*, *determinization* and *minimization* to combine all the knowledge sources used in speech recognition and optimize into a single compact search network [7]. This approach works well for certain types of transducers, but presents some problems related to the applicability of *determinization* and *weight-pushing* with more general transducers [8]. In this respect, Turkish morphology presents a problem, since the number of ambiguities is infinite and the cycle-ambiguous finite-state transducer of the morphological parser is not determinizable. Still, we can apply the local determinization algorithm for lo-

[6] We built a finite-state transducer based pronunciation lexicon similar to [37] and extended the phone set, however it did not lead to performance improvement possibly due to a small number of Turkish words with exceptional pronunciation.

cally optimizing the search network using the *grmlocaldeterminize* utility from *AT&T Grammar Library* [88]. The experimental results show that this approach works well.

4.3. Discriminative Reranking with Perceptron

The introduction of arbitrary and global features into the generative models results in difficulty due to the finite-state nature of these models. Therefore, the common approach in NLP research has been to use a baseline generative model to generate ranked n-best candidates, which are then reranked by a rich set of local and global features [18,34].

The perceptron algorithm has been successfully applied to various NLP tasks for ranking or reranking hypotheses [1,16,18,20,34]. The perceptron has shown significant improvements for discriminative language modeling for Turkish using linguistic and statistically derived features [89]. It also gives the best performance for morphological disambiguation of Turkish text using morpholexical features [72]. The characteristics like simplicity, fast convergence, and easy incorporation of arbitrary local and global features make the perceptron algorithm very attractive for discriminative training of linear models. In this section, we introduce a variant of the perceptron, WER-sensitive perceptron, which is better suited to rerank n-best speech recognition hypotheses.

4.3.1. The WER-sensitive Perceptron Algorithm

The perceptron is a linear classifier [19]. The perceptron algorithm tries to learn a weight vector that minimizes the number of misclassifications. Figure 4.1 shows a variant of the perceptron algorithm, *WER-sensitive perceptron*, formulated as a multiclass classifier which is very similar to the the averaged perceptron [1,20]. The algorithm estimates a parameter vector $\bar{\alpha} \in \Re^d$ using a set of training examples $(x_i, y_i) : 1 \leq i \leq n$. The function $\mathbf{GEN}$ enumerates a finite set of candidates $\mathbf{GEN}(x) \subseteq Y$ for each possible input x. The representation $\mathbf{\Phi}$ maps each $(x, y) \in X \times Y$ to a feature vector $\mathbf{\Phi}(x, y) \in \Re^d$. The learned parameter vector $\bar{\alpha}$ can be used for mapping unseen inputs $x \in X$ to outputs $y \in Y$ by searching for the best scoring output, i.e.

```
input set of training examples ${(x_i, y_i) : 1 \leq i \leq n}$
input number of iterations $T$
$\bar{\alpha} = 0$, $\bar{\alpha}_{sum} = 0$
for $t = 1 \dots T$, $i = 1 \dots n$ do
    $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \bar{\alpha}$
    $\bar{\alpha} = \bar{\alpha} + \Delta(y_i, z_i)(\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i))$
    $\bar{\alpha}_{sum} = \bar{\alpha}_{sum} + \bar{\alpha}$
end for
return $\bar{\alpha}_{avg} = \bar{\alpha}_{sum}/(nT)$
```

Figure 4.1. The WER-sensitive perceptron algorithm.

$\arg\max_{z \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, z) \cdot \bar{\alpha}$. The given algorithm can also be used to rank the possible outputs for an input x by their scores, $\mathbf{\Phi}(x, z) \cdot \bar{\alpha}$.

The algorithm makes multiple passes (denoted by T) over the training examples. For each example, it finds the highest scoring candidate among all candidates using the current parameter values. If the highest scoring candidate is not the correct one, it updates the parameter vector $\bar{\alpha}$ by the difference of the feature vector representation of the correct candidate and the highest scoring candidate. This way of parameter update increases the parameter values for features in the correct candidate and downweights the parameter values for features in the competitor. For the application of the model to the test examples, the algorithm calculates the "averaged parameters" since they are more robust to noisy or inseparable data [1]. The averaged parameters $\bar{\alpha}_{avg}$ are calculated by summing the parameter values for each feature after each training example and dividing this sum by the total number of updates. We define X, Y, x_i, y_i, $\mathbf{GEN}$, and $\mathbf{\Phi}$ of the perceptron algorithm in a reranking setting of ASR hypotheses as follows:

- X is the set of all possible acoustic inputs.
- Y is the set of all possible strings, $\sum^*$, for a vocabulary $\sum$ which can be a set of words, sub-words, or morpholexical units of the generative language model.
- Each x_i is an utterance - a sequence of acoustic feature vectors. The training set

contains n such utterances.

- $\mathbf{GEN}(x_i)$ is the set of alternate transcriptions of x_i as output from the speech decoder. Although the speech decoders can generate lattices which encode alternate recognition results compactly, we prefer to work on n-best lists for efficiency reasons and very small performance gains with the lattices.
- y_i is the member of the $\mathbf{GEN}(x_i)$ with the lowest word error rate with respect to the reference transcription of x_i. Since there can be multiple transcriptions with the lowest error rate, we take y_i to be the one with the best score among them.
- Each component $\Phi_j(x, y)$ of the feature vector representation $\mathbf{\Phi}(x, y) \in \Re^d$ holds the number of occurrences of a feature or indicates the existence of a feature. For instance one of the features can be defined on part of speech tags of the words as follows:

  $\Phi_1(x, y) =$ number of times an *adjective* is followed by a *noun* in y.
- The expression $\mathbf{\Phi}(x, y) \cdot \bar{\alpha}$ denotes the inner product $\sum_{j=1}^{d} \Phi_j(x, y)\alpha_j$, where α_j is the j^{th} component of the parameter vector $\bar{\alpha}$.
- The zeroth component $\Phi_0(x, y)$ can represent the log-probability of y (weighted sum of the baseline language and acoustic model scores) in the lattice output from the baseline recognizer for utterance x. We experimented with the perceptron algorithm where this baseline score can be included or omitted in training. During testing, the baseline score as the zeroth feature is always included. The corresponding weight α_0 for $\Phi_0(x, y)$ is fixed and optimized on a held-out set.

With this setting, the perceptron algorithm learns an averaged parameter vector $\bar{\alpha}_{avg}$ that can be used to choose the transcription y having hopefully the least number of errors for an utterance x using the following function:

$$F(x) = \operatorname*{arg\,max}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \bar{\alpha}_{avg}$$

The WER-sensitive perceptron algorithm is obtained by defining a better loss function tailored for reranking ASR hypotheses. The loss function of the averaged perceptron [1] algorithm can be written as follows:

$$L(\bar{\alpha}) = \sum_{i=1}^{n} [\![\bar{\alpha} \cdot \mathbf{\Phi}(x_i, z_i) - \bar{\alpha} \cdot \mathbf{\Phi}(x_i, y_i)]\!]$$

where $[\![x]\!] = 0$ if $x < 0$ and 1 otherwise. We can define a better loss function which is based on the total number of extra errors we do by selecting the candidates with higher WER rather than the best candidates. Then minimizing the loss function corresponds to minimizing the WER of the reranker. We define the word error rate sensitive loss function as follows:

$$L(\bar{\alpha}) = \sum_{i=1}^{n} \Delta(y_i, z_i) [\![\bar{\alpha} \cdot \mathbf{\Phi}(x_i, z_i) - \bar{\alpha} \cdot \mathbf{\Phi}(x_i, y_i)]\!]$$

where the loss function $\Delta(y_i, z_i)$ for each example x_i is defined as the difference of edit distances of z_i and y_i with the reference transcription of x_i.

The gradient of the loss function can be found by the stochastic gradient descent as $\lambda\Delta(y_i, z_i)(\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i))$, where λ is the learning rate. For the experiments, we set the λ to 1, for which we found the algorithm to converge well. We also provide a proof of convergence for the WER-sensitive perceptron algorithm for linearly separable training sequences in the appendix.

Note that a loss-sensitive perceptron algorithm has been proposed for reranking speech recognition output in [22]. Although this work is similar in using edit distance as a loss function, they use it for scaling the margin to ensure that hypotheses with a large number of errors are more strongly separated from the members of the set of lowest error (optimal) hypotheses. They also update the weight vector using features from optimal and non-optimal set of hypotheses that violate the scaled margin.

4.4. Experiments

This section gives experimental results for the application of proposed generative and discriminative language models to a Turkish broadcast news transcription task.

4.4.1. Broadcast News Transcription System

The automatic transcription system uses hidden Markov models (HMMs) for acoustic modeling and WFSTs for model representation and decoding. The HMMs are decision-tree state clustered cross-word triphone models with 10843 HMM states and each state is a Gaussian mixture model (GMM) having 11 mixture Gaussian densities with the exception of silence model having 23 mixtures. The model has been trained on 188 hours of acoustic data from the Boğaziçi broadcast news (BN) database [3,38]. Separate from the training data, disjoint held-out (3.1 hours) and test (3.3 hours) data sets are used for parameter optimization and final performance evaluation, respectively.

The language models are trained using two text corpora. The larger corpus is the NewsCor corpus (184 million words) described in Section 3.3 and acts as a generic corpus collected from news portals. The other one is the BN corpus (1.3 million words) and it contains the reference transcriptions of BN database and acts as in-domain data. The generative language models are built by linearly interpolating the language models trained on these corpora. The interpolation constant is chosen to optimize the perplexity of held-out transcriptions. The baseline n-gram language models are estimated with interpolated Kneser-Ney smoothing and entropy-based pruning using the SRILM toolkit [39]. The discriminative models are trained using only the BN corpus. The speech recognition experiments are performed by using the AT&T DCD library [7] . This library is also used for the composition and optimization of the finite-state models to build the search network for decoding.

[7] http://www.research.att.com/~fsmtools/dcd/

4.4.2. Generative Language Models

We evaluated the performance of the proposed morpholexical language model against the word and morph models on the broadcast news transcription task. We experimented with two different morpholexical language models with modeling units of lexical-grammatical morpheme and lexical stem+ending.

For the experimental set-up of the word and morph based models, we used the settings in the previous studies of [3, 38]. The vocabulary size of 200K and n-gram order of 3 were chosen for the word based model to balance the trade-off between recognition performance and computational cost increasing with the model complexity. The OOV rate of the test set with the 200K word vocabulary is 2%. For the morph based model, we employed the best performing method of marking non-initial morphs with "-", which is used to locate the word boundaries for the purpose of conversion from morph sequences of recognition results to word sequences. This method increased the vocabulary size from 50K to 76K. The OOV rate of the test set with the 76K morph vocabulary is 0%, since the letters are also included in the morphs lexicon. The morph based experiments were conducted with 4-gram language models. To build the morpholexical language models, the text corpora were morphologically parsed and disambiguated to get the lexical-grammatical morpheme and lexical stem+ending representations of corpora. The lexicon of the morphological parser contains about 88K symbols. The OOV rate of the morphological parser on the test set is about 1.3%. The lexical-grammatical morpheme representation resulted in about 175K symbols. The lexical stem+ending representation resulted in about 200K symbols. For both morpholexical units, the n-gram order of 4 was chosen. The morpholexical search networks were built using the lexical transducer of the morphological parser and the weighted finite-state automata representation of the morpholexical language models as explained in Section 4.2. The search network was optimized using local determinization from GRM library [88].

Figure 4.2 shows the word error rate versus real-time factor for the word, morph, morpheme, lexical stem+ending, and surface form stem+ending models for the first-

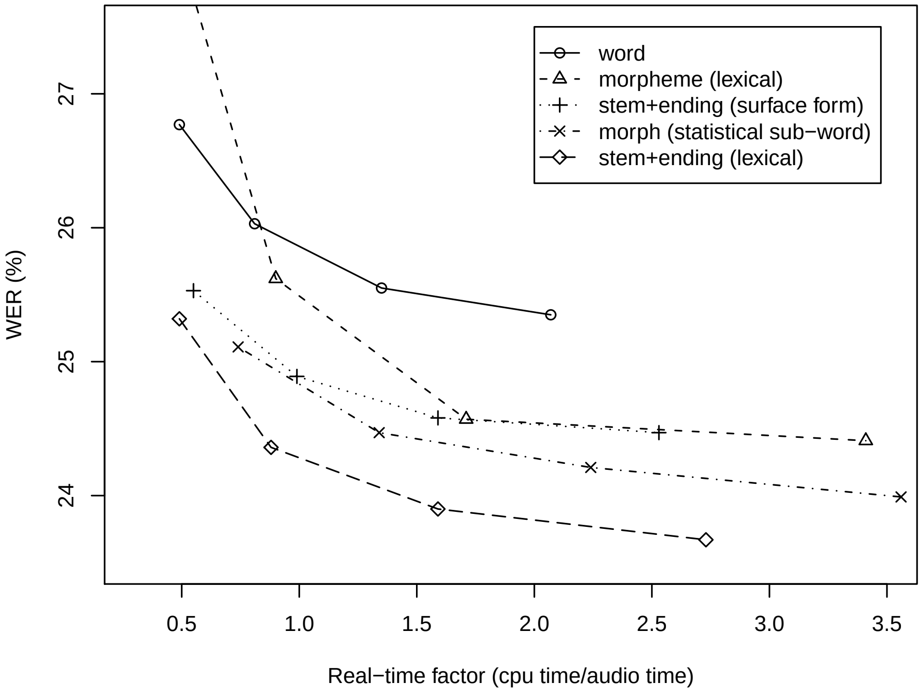


Figure 4.2. Word error rate for the first-pass versus real-time factor obtained by changing the pruning beam width.

pass. Note that since the morpholexical models output recognition results in morphological representation, we use the inverse of lexical transducer of morphological parser as a word generator to convert them to words to calculate the WERs. Since the language models are pruned for computational reasons when building the optimized search networks of first-pass recognition, we rescore the output lattices with unpruned language models as a second-pass. Table 4.2 shows the recognition results for the rescoring of n-best hypotheses for all the models at ∼1.5 real-time factor. As can be seen from the first-pass and second-pass results, the sub-word and sub-lexical models perform significantly better than the word-based model. As a morpholexical language model, the lexical stem+ending model has the best performance.

4.4.3. Effectiveness of Morphotactics and Morphological Disambiguation

In this section, we give experimental results showing the effect of morphotactics and morphological disambiguation on speech recognition performance using the lexical stem+ending model. Figure 4.3 shows the word error rate of the first-pass speech

Table 4.2. Results for rescoring with unpruned language models.

Model	**WER (%)**	**Rescore WER (%)**
word	25.6	23.4
morpheme	24.6	22.2
stem+ending (surface)	24.6	22.0
morph	24.5	22.4
stem+ending (lexical)	23.9	21.6

recognition at various real-time factors using four different language models. The baseline model is the lexical stem+ending model with the correct morphotactics and morphological disambiguation. First, we experimented with the morphotactics. The *stem+ending:no-mt* model represents the experiment where the morphotactics component of the lexical transducer allows any ordering of the morphemes. Second, we tested the effectiveness of doing morphological disambiguation on the language model text corpus. The *stem+ending:no-disamb* model represents the case where the morphological disambiguation is replaced with choosing the morphological parse with the least number of morphemes. The final model *stem+ending:no-mt-no-disamb* shows the cumulative effect for the absence of morphotactics and morphological disambiguation. It is clear that morphotactics is effective in reducing the error rate. This result shows that morphotactics is successful in constraining the search space to valid morpheme sequences. Besides, this figure shows that morphological disambiguation also improves speech recognition performance. We can conclude that morphological disambiguation improves the prediction power of morpholexical language model. The absence of morphotactics and disambiguation together has a larger impact on recognition performance.

4.4.4. Effect of Pronounciation Modeling

Most of the words in Turkish have one-to-one mapping between graphemics and pronunciation. However, there are quite a number of exceptional subtle phenomena in loan words, and in morphophonology, such as vowel length alternations, which are distinguished in phonetic transcription rather than orthography. For instance, there

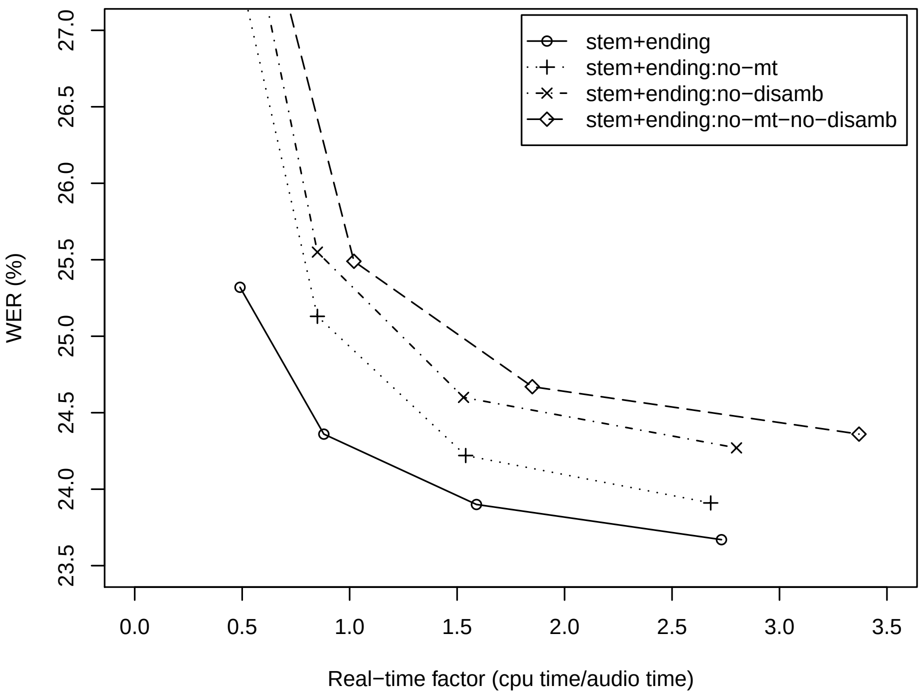


Figure 4.3. Effects of morphotactics and morphological disambiguation for the lexical stem+ending model.

are some homographs which can be morphologically parsed in multiple ways having different pronunciations, such as *karın*. Oflazer describes a finite-state transducer implementation for a Turkish pronunciation lexicon [37]. However, to the best of our knowledge, there has been no study to evaluate the effect of pronunciation modeling for Turkish ASR, where the orthographical symbols in the alphabet have been generally used for acoustic and pronunciation modeling.

To study the effect of pronunciation, we built a finite-state pronunciation lexicon for Turkish. The architecture of the system is quite different than [37]. We built a pronunciation lexicon for exceptional root words (about 3700) which are mostly loan words and do not have a direct mapping to orthography. We also compiled a modified two-level rule transducer that operates on phonemes rather than graphemes. The resulting morphophonemics transducer also handles exceptional morphophonology such as lengthening of vowels. As the phoneme set, we used a proprietary phoneme set which is based on SAMPA standard[8] , but the phonetic changes are reflected on

[8] http://www.phon.ucl.ac.uk/home/sampa/turkish.htm

the vowels. This makes morphophonological alternations easier to handle in two-level morphology. We build the lexical transducer of the pronunciation lexicon by composing the morphophonemics transducer and lexicon/morphotactics transducer updated with the exceptional root pronunciations: $MP \circ MT$. Then the pronunciation lexicon can be constructed by composing the inverse of this transducer with the lexical transducer of the morphological parser: $MG \circ MT \circ MT_{new}^{-1} \circ MP^{-1}$.

We used this pronunciation lexicon to build a new acoustic model and experimented in broadcast news transcription. As an early investigation, we can confirm that using phonemes instead of graphemes did not improve the ASR performance. This can be expected since exceptional words are mostly loan words and their frequency is quite low. It is also quite possible that acoustic model distinguishes between most of them since it employs context dependent phones.

4.4.5. Discriminative Reranking of ASR Hypotheses

The speech decoder generates word, sub-word or morpheme lattices depending on the units of the language model used in the first pass. Then, we extract an n-best list of hypotheses from these lattices which are ranked by the combined score obtained from the language and acoustic model. The resulting n-best hypotheses are reranked with a discriminative linear model trained with the perceptron algorithm using the features extracted from the hypotheses.

In the reranking experiments, we used the experimental setup of Arısoy [3]. The n-best hypotheses for all systems are generated by decoding the acoustic training data with the corresponding generative model. The acoustic model trained on all the utterances in the training data is used to decode all the utterances. However, in language modeling, 12-fold cross validation is employed to prevent over-training of the discriminative model. This is done by decoding utterances in each fold with a fold-specific language model which is built by interpolating the generic language model trained on NewsCor corpus with the in-domain language model trained with the reference transcriptions of the utterances in the other 11 folds. The same interpolation constant -

0.5 - is used for building fold-specific language models of all systems. 200K word, 76K morph and 200K lexical stem+ending units of vocabulary were employed while building 3-gram word, 4-gram morphs, and 4-gram lexical stem+ending models, respectively. Since the n-gram language models are pruned for computational reasons, the lattices generated in the first-pass at $\sim$1.5 real-time factor are rescored with unpruned language models.

The reranking models are trained both with the WER-sensitive perceptron algorithm of Figure 4.1 and the original averaged perceptron algorithm. We also carried out experiments to see the effect of using baseline score in discriminative training. The 50-best hypotheses extracted for each utterance from the rescored lattices are used for the training and reranking. The number of iterations of the algorithm and the weight α_0 used for scaling the hypothesis score from the first-pass are optimized on a held-out set.

The final reranking results are given on a test set in Table 4.3. The WER-sensitive perceptron algorithm shows consistent improvements for all the models on the test set. Dismissing the first-pass score in the training of the standard perceptron degrades reranking performance on the test set. In contrast, it seems generally better to dismiss the first-pass score for the WER-sensitive perceptron. This is important since we might not have these scores if we want to train the discriminative model in an unsupervised manner where we don't have the transcribed acoustic data.

To evaluate the effectiveness of the WER-sensitive perceptron algorithm, we carried out significance tests using the NIST MAPSSWE test for the morph model where the algorithm seems to make significant difference. The WER-sensitive perceptron without the baseline score in training gives the best word error rate of 21.5% for the morph model as can be seen in Table 4.3. The performance improvement of this model over three other configurations is significant at the levels of $p = 0.048$, $p = 0.004$, and $p < 0.001$ with respect to the increasing word error rates of the configurations. The stem+ending model performs significantly better than the word and morph model for all the configurations ($p < 0.001$).

Table 4.3. Discriminative reranking results with the perceptron using unigram features.

WER sensitive			–	–	✓	✓
Baseline score in training			–	✓	✓	–
Model	oracle	1-best	reranking			
word	15.0	23.4	23.2	23.0	22.9	23.0
morph	13.9	22.4	21.9	21.8	21.7	21.5
stem+ending	13.7	21.6	21.1	20.9	20.9	20.8

Although we used the unigram features for all the models in the reranking experiments, Arısoy [89] has shown that using richer linguistic and statistically derived features further improves the reranking performance.

4.5. Discussion

We first introduced the morpholexical language model which is a morphological sub-lexical n-gram language model. Second, we showed that we can build a morphology-integrated search network for ASR using a morpholexical language model and the lexical transducer of the morphological parser in the finite-state transducer framework. This proposed approach is superior to word n-gram models in the following aspects:

- The vocabulary is unlimited since the modeling units are sub-lexical units.
- It alleviates the OOV and vocabulary growth problem. The OOV rate is effectively reduced to about 1.3% on the test set. For comparison, the 200K word model has about 2% OOV rate.
- Using lexical units alleviates data sparsity problem.
- Lexical stem+ending model gives the best results, and it improves the WER over word model by 1.8% absolute.

Besides, it is superior to statistical sub-word (morph) models in some other aspects:

- The modeling units as being lexical and grammatical morphemes provide a linguistic approach.
- The linguistic approach enables integration with other finite-state models like pronunciation lexicon.
- The morphology-integrated search network only allows valid word sequences thanks to the morphotactics.
- The morphological features can be further exploited in a rescoring or reranking model.
- Lexical stem+ending model improves the WER over morph model by 0.8% absolute.

The experimental results show that lexical stem+ending model as a morpholexical language model outperforms all the other models significantly. We also show that morphotactics and morphological disambiguation are effective for better language modeling.

Third, we presented a variant of the perceptron algorithm, WER-sensitive perceptron, for discriminative training of reranking models. The experimental results show that this algorithm is better for reranking speech recognition hypotheses. The reranking WER for the lexical stem+ending model is lower by 2.2% and 0.7% absolute than word and morph models respectively.

Although, the language models and techniques in this work have been developed and applied for Turkish speech recognition, they can be applied for other morphologically rich languages such as Arabic, Finnish, and Czech and in other language processing applications. We believe that using grammatical sub-lexical units in language modeling can be even more beneficial for other applications especially for machine translation.

5. ON-THE-FLY LATTICE RESCORING FOR REAL-TIME ASR

A speech recognition lattice is a weighted directed acyclic graph where each path from the start state to a final state represents an alternative transcription hypothesis, weighted by its recognition score for a given utterance [2]. For large vocabulary speech recognition, the n-gram language models are often pruned or the order of the language model is lowered for computational reasons when building the optimized search networks of the first-pass recognition. Then the output word lattices from the first-pass are rescored offline with unpruned language models or higher order n-grams. However, the real-time speech recognition systems such as the automatic closed captioning system by Saraçlar *et al.* [90] that require low-latency cannot benefit from the lattice rescoring as the latency increases with the size of the output lattice [91].

In this chapter, we propose an algorithmic framework for rescoring lattices on-the-fly. The lattice generation method we employ is based on the lattice generation algorithm of Ljolje *et al.* [2]. Although the algorithm we use for rescoring recognition hypotheses is similar to the on-the-fly hypothesis rescoring algorithm of Hori *et al.* [92], there are major differences. First of all, two methods concentrate on different problems. While we employ a hypotheses rescoring method for producing rescored lattices on-the-fly, Hori *et al.* [92] uses a similar method in an on-the-fly composition algorithm setting to achieve fast and memory-efficient decoding in extremely large vocabulary continuous speech recognition. They decompose the search network into two transducer groups and a Viterbi search is performed based on the first transducer, whereas the second transducer is used to rescore the hypotheses and the updated scores are used in the Viterbi search. Second, we propose a more general lattice rescoring framework in terms of enabling more general rescoring models rather than setting it up as a composition of two finite-state models as in [92].

5.1. WFST-based Speech Decoding

The weighted finite-state transducers (WFSTs) are weighted directed graphs in which each arc a has a source state $S(a)$, a destination state $D(a)$, an input label $I(a)$, an output label $O(a)$, and a weight $P(a)$. The WFSTs provide a unified framework for representing different knowledge sources in ASR systems, e.g., hidden Markov models (HMMs), context-dependent dependency networks, pronunciation lexicons, and n-gram language models [6].

In the WFST framework, the speech recognition problem is treated as a transduction from input speech signal to a word sequence. The various knowledge sources are represented as WFSTs. A typical set of knowledge sources consists of a context-dependency network C transducing context-dependent phones to context-independent phones, a lexicon L mapping context-independent phone sequences to words, and an n-gram language model G assigning probabilities to word sequences. The composition of these models $C \circ L \circ G$ results in an all-in-one search network that directly maps context-dependent phone (corresponding to an HMM) sequences to weighted word sequences, where weights can be combinations of pronunciation and language model probabilities. The WFST also offers finite-state operations such as *composition*, *determinization* and *minimization* to combine all these knowledge sources into an optimized all-in-one search network.

The decoding for the best path in the resulting network is a single-pass Viterbi search. In this work, we implemented a WFST-based Viterbi speech decoder to experiment with the on-the-fly lattice rescoring using the OpenFst library [75].

5.1.1. One-Best Decoding

The decoding for the best path of arcs $\mathbf{a} = a_1...a_n$ from the initial state to a final state in a search network T given acoustic feature vectors $\vec{x}$ for a utterance can be formulated as in [2]:

$$\max_{\mathbf{a}} P(\vec{x}[0,\tau],\mathbf{a}) = \max_{\mathbf{a},\mathbf{t_1},\ldots,\mathbf{t_{n-1}}} \prod_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{P}(\tilde{\mathbf{x}}[\mathbf{t_{i-1}},\mathbf{t_i}] \mid \mathbf{I}(\mathbf{a_i}))\mathbf{P}(\mathbf{a_i})$$

The probability $P(\vec{x}[t_{i-1},t_i] \mid I(a_i))$ represents the likelihood assigned by the acoustic model for the context-dependent phone $I(a_i)$ when applied to the feature vectors $\vec{x}$ for the time interval t_{i-1},t_i. The probability $P(a_i)$ is the language model probability for the arc a_i in T.

We can formulate this equation in terms of the best scoring path to each state of T at a given time instant for the Viterbi algorithm. Let $B(s)$ be the set of all paths $a_1...a_k$ in T from the initial state to state s and define that best scoring path as:

$$\alpha(s,t) = \max_{a\in B(s),t_1,\ldots,t_{k-1}} \prod_{i=1}^{k} P(\vec{x}[t_{i-1},t_i] \mid I(a_i))P(a_i)$$

Then:

$$\max_{\mathbf{a}} P(\vec{x}[0,t],\mathbf{a}) = \max_{\mathbf{s}} \alpha(\mathbf{s},\mathbf{t}) =$$

$$\max_{D(a)=s} P(a)\left[\max_{t'<t} P(\vec{x}[t',t] \mid I(a))\alpha(S(a),t')\right] \tag{5.1}$$

In Equation 5.1, the Viterbi decoding of the best scoring path at a time instant is expressed as two nested (max) loops: the outer loop considers each possible active arc a ending in state s, and the inner loop finds the optimal start time t' for a by combining the acoustic likelihood of a between t' and t with the best path score from the start to state $S(a)$ at time t'.

```
Each state of lattice $L$: $(t, s) \rightarrow$ a time frame $t$ and a state $s$ in $T$
for $t = t_0$ to $\tau$ do
  for each active arc $a$ do
    $t' \leftarrow \arg\max_{t'<t} P(\vec{x}[t', t] \mid I(a))P(a)\alpha(S(a), t')$
    NewArc from: $(t', S(a))$ to: $(t, D(a))$ input: $I(a)$ output: $O(a)$
    weight: $P(\vec{x}[t', t] \mid I(a))P(a)$
  end for
end for
```

Figure 5.1. Lattice generation algorithm of Ljolje *et al.* [2]

5.1.2. Lattice Generation

The lattice generation method for on-the-fly lattice rescoring algorithm that we propose is based on the the lattice generation algorithm of Ljolje *et al.* [2], which is shown in Figure 5.1. Their lattice generation algorithm involves no extra computation over the normal Viterbi algorithm other than the negligible time needed to add states and arcs into the transducer lattice as it is being constructed. Each state of a phone-to-word transducer lattice L corresponds to a pair (t, s) of a time frame in the recognition and a state from the recognition transducer T. The initial state is the pair of utterance start time 0 and the start state of T. The final states are the pairs consisting the utterance end time τ and a final state from T. If, during the Viterbi recursion of Equation 5.1, we have identified the optimal start time t' for arc a ending in state $D(a)$ at time t, then a corresponding arc is added to L from state $(t', S(a))$ to state $(t, D(a))$. If necessary, state $(t, D(a))$ was first created ; state $(t', S(a))$ must already have been in L by induction. The new arc has input label $I(a)$, output label $O(a)$, and weight $P(\vec{x}[t_{i-1}, t_i] \mid a)P(a)$. All this information is readily available from the Viterbi recursion. We store the index for the pair $(t, D(a))$ in the decoder active state data structure for $D(a)$ at the current time, and we store the index for the pair $(t', S(a))$ in the decoder active arc data structure for a. The generated phone-to-word transducer lattice can also be converted to a word lattice and pruned relative to the best scoring path through the entire lattice as explained in [2].

5.2. Lattice Rescoring

In this section, we describe the lattice rescoring method with composition as the baseline and the proposed on-the-fly rescoring algorithm.

5.2.1. Lattice Rescoring with Composition

The word lattices output from a speech recognizer generally contain both the acoustic model and the language model scores for the hypotheses as explained in the previous section. For rescoring the lattices offline, the scores from the language model of the first-pass for a transcription hypothesis is subtracted from that hypothesis' lattice score and the resulting weighted finite-state automaton is intersected with the rescoring language model automata. However, with a simple modification to the lattice generation algorithm of [2] given in Section 5.1.2 - by assigning only the acoustic model score $P(\vec{x}[t_{i-1}, t_i] \mid a)$ to a new lattice arc and omitting the language model score $P(a)$ - we can get rid of the score subtraction step. Then we can just intersect the output lattice with the better language model for rescoring. For short utterances, this method of rescoring is very effective and has a very small latency. However, for real-time speech recognition systems that require decoding long utterances as in Saraçlar *et al.* [90], this method is not feasible since the memory for generating and storing the lattice increases rapidly and the composition with large lattices leads to significant latency. This method also requires that the rescoring model could be represented as weighted finite-state automata, which may not be the case, for instance, for the discriminative language models with complex feature sets.

In the ASR experiments, we implemented this method as a baseline to compare with the on-the-fly rescoring method.

5.2.2. On-the-fly Lattice Rescoring

For on-the-fly lattice rescoring, we need an algorithm for lattice generation and rescoring recognition hypotheses. The lattice generation algorithm is based on the

```
Each state of lattice $R$: $(t, s, h) \rightarrow$ a time frame $t$, a state $s$ in $T$ and an $n$-gram
history $h$ for a path arriving to state $s$ at time instant $t$
for $t = t_0$ to $\tau$ do
  for each active arc $a$ do
    $t' \leftarrow \arg\max_{t'<t} P(\vec{x}[t', t] \mid I(a))P(a)\alpha(S(a), t')$
    for each $n$-gram history $h'_{t'}(a)$ in arc $a$ do
      NewArc from: $(t', S(a), h'_{t'}(a))$ to: $(t, D(a), h_t(a))$ input: $I(a)$ output:
      $O(a)$ weight: $P(\vec{x}[t', t] \mid I(a))P(O(a)|h'_{t'}(a))$
    end for
  end for
end for
```

Figure 5.2. On-the-fly Lattice Rescoring algorithm.

algorithm given in Section 5.1.2. The recognition hypothesis rescoring methodology is conceptually similar to the on-the-fly hypothesis rescoring algorithm of Hori *et al.* [92], however the motivation and implementation of hypothesis rescoring is different.

The on-the-fly lattice rescoring algorithm is shown in Figure 5.2. Each state of a phone-to-word rescored transducer lattice R corresponds to a tuple (t, s, h) of a time frame in the recognition, a state from the recognition transducer T, and an n-gram history for a path arriving to state s at time instant t. Since there may be multiple paths arriving at a state at the same time during decoding with possibly different n-gram histories, we need to keep track of n-gram word histories for active arcs and states. The initial state is the tuple of utterance start time 0, the start state of T, and the sentence start symbol <*s*>. The final states are the tuples consisting the utterance end time τ, a final state from T, and the sentence end symbol <*/s*>. If, during the Viterbi recursion of Equation 5.1, we have identified the optimal start time t' for arc a ending in state $D(a)$ with an n-gram history h at time t, then a corresponding arc is added to R from state $(t', S(a), h)$ to state $(t, D(a), h')$ with updated history h'. If the output label $O(a)$ of that arc a is ϵ, h' is the same with h. If not, then the new history h' is formed by dropping the oldest word and appending the output label $O(a)$ to the history. If necessary, state $(t, D(a), h')$ was first created ; state $(t', S(a), h)$ must

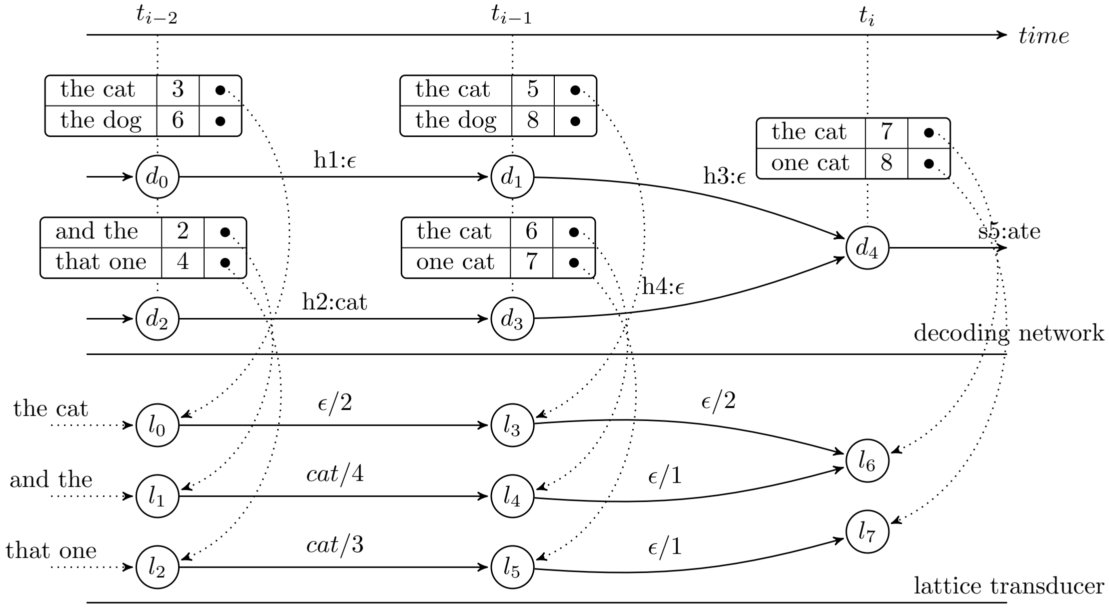


Figure 5.3. Hypotheses and associated lattice rescoring information during decoding.

already have been in L by induction. The new arc has input label $I(a)$, output label $O(a)$, and weight $P(\vec{x}[t_{i-1}, t_i] \mid a)P(O(a)|h)$. $P(O(a)|h)$ is the language model score assigned by the rescoring language model to the current n-gram. If the output symbol is ϵ, it is taken as 1 to use the acoustic model score. All this information is readily available from the Viterbi recursion. We store the index for the tuple $(t, D(a), h')$ in the decoder active state data structure for $D(a)$ at the current time, and we store the index for the pair $(t', S(a), h)$ in the decoder active arc data structure for a.

5.2.3. Implementation Details

Figure 5.3 shows a partial decoding process of the Viterbi search progressing in time. The upper part of the figure shows the states activated at a time instant t_i. Each active state stores a pointer to a list of rescoring tokens as shown above each state. In the rescoring token list, we store the forward acoustic score of an active state at the creation time of the list. This score is used to update the rescoring token scores with the accumulated acoustic score. During the Viterbi decoding, we store a list of maximum N rescoring tokens in each active state, where each token represents a word trace (path) from the rescoring language model. A rescoring token contains the current n-gram history, the score for the current rescoring state, and a pointer to a lattice state that is being generated on-the-fly. When two hypotheses meet at the same active state, the rescoring token lists are merged and the maximum N tokens with the N-best scores are kept.

During decoding, each active arc has an associated HMM with a number of active HMM states. In Figure 5.3, the active HMM states are not shown for clarity, but the decoding process is very similar in the internal HMM states. When two hypotheses meet at the same HMM state in an active arc, only the rescoring token list having the token with the best score is retained. The pointers to the rescoring token lists are shared pointers with reference counting. Therefore in the internal Viterbi decoding of HMM states within an active arc, we can just propagate the pointers without copying the lists.

Each rescoring token in a rescoring token list stores a unique n-gram history to keep track of the state information in the rescoring language model. In Figure 5.3, we use a 3-gram rescoring language model, therefore we only need to store bigram word histories. This way of implementation provides a more general framework for rescoring, since we can also use rescoring models that cannot be represented as finite-state transducers. In the experiments, we used a 3-gram language model and used the SRILM toolkit [39] to train language models and assign probabilities to n-grams on-the-fly. However, it is also possible to use other language models such as a discriminative language model to assign scores. Note that we can also use discriminative models with acoustic, duration and language model based features such as [93] in the first pass by simple modifications to the proposed lattice rescoring algorithm.

As indicated above, we generate the word lattices as in Ljolje *et al.* [2], however one can also use the method by Saon *et al.* [29]. The lower part of Figure 5.3 shows the rescored lattice being generated on-the-fly. Each rescoring token has a pointer to a lattice state as shown by dashed arcs from decoding network to the lattice network. When an active arc in the decoding network is expanded to a new state, that state is activated and the rescoring token list propagated from the active arc is updated with the new lattice pointers while adding new lattice arcs and states.

5.3. Experiments

We evaluated the performance of the rescoring algorithms on a Turkish broadcast news transcription task. The acoustic model uses hidden Markov models (HMMs) trained on 188 hours of broadcast news speech data [38]. In the acoustic model, there are 10843 triphone HMM states and 11 Gaussians per state with the exception of the 23 Gaussians for the silence HMM. The test set contains 3.1 hours of speech data that has been pre-segmented into short utterances (2410 utterances and 23038 words). We used the geometric duration modeling in the decoder.

The speech decoder is our implementation of a WFST-based decoder using the OpenFst library for finite-state operations and model representations [75]. We implemented the lattice generation algorithm of Ljolje *et al.* [2] to produce the lattices. The proposed algorithm for on-the-fly lattice rescoring has also been implemented in the decoder.

The text corpora that we used for building n-gram language models are composed of about 200 million-words BOUN NewsCor corpus collected from news portals in Turkish and 1.3 million-words text corpus (BN Corpus) obtained from the transcriptions of the Turkish Broadcast News speech database [38]. The language model of the decoding network is a 3-gram language model with a vocabulary size of 200K words. This model is estimated by linearly interpolating two language models trained over the BOUN NewsCor corpus and the BN corpus to reduce the effect of out-of-domain data. The language model trained on the BOUN NewsCor corpus is pruned due to high memory requirements while building the optimized search network using the SRILM toolkit [39]. The language model used for lattice rescoring experiments use the unpruned language models with the same n-gram order of 3.

We give speech recognition results for three systems. All the systems are single-pass systems. In the first system, the decoder generates a lattice using the pruned search network without any rescoring. The second system uses the pruned search network to generate a lattice containing only acoustic model scores and the resulting

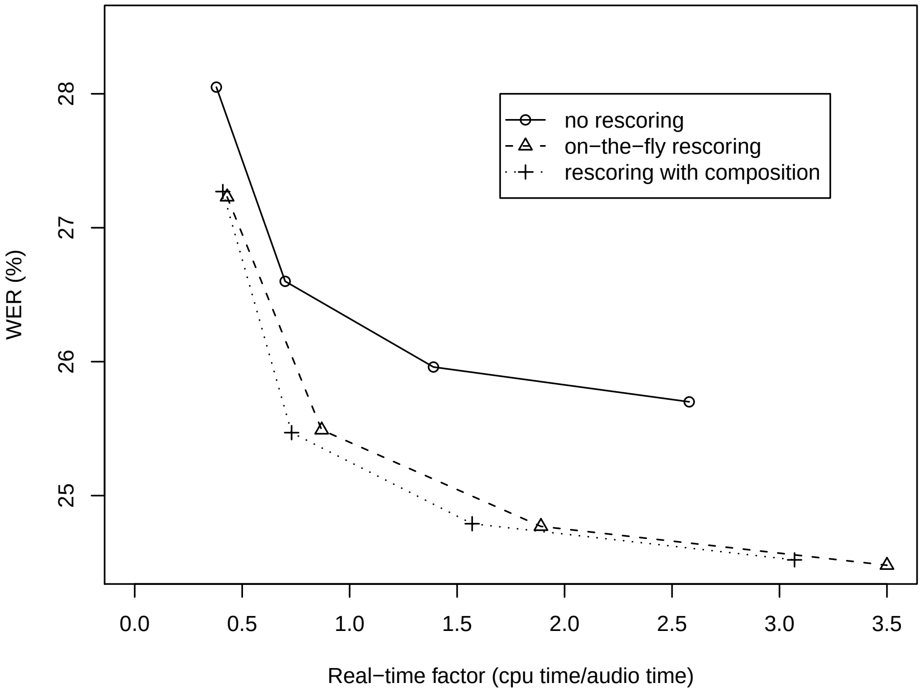


Figure 5.4. Word error rate versus real-time factor obtained by changing the pruning beam width.

lattice is composed with the unpruned language model to rescore it as soon as the recognition for each utterance ends. The third system uses our proposed on-the-fly lattice rescoring method.

Figure 5.4 shows the word error rate versus run-time factor obtained by varying prune beam widths from 9 to 12 for three systems. As expected rescoring with unpruned language models improves the accuracy. Since the utterances in our test set are short, the baseline rescoring method has not a significant effect on run-time for small beam-widths. The on-the-fly rescoring method achieves about the same accuracy with the baseline rescoring method. The proposed lattice rescoring method can generate rescored lattices with less than 20% increased computation over standard lattice generation. However, the accuracy improvement for a real-time system (i.e. the real-time factor 1 in the graph) is very significant for the on-the-fly rescoring method without increasing the latency of the baseline system with no rescoring.

6. CONCLUSIONS

In this thesis, we presented our approaches for solving the problems in speech and language processing of Turkish associated with its productive morphology. We focused on improving statistical language modeling of Turkish. However, the proposed methodologies can be applied to other languages having agglutinative or inflectional morphology, such as Arabic, Czech, Finnish, and Korean. The developed language modeling approaches have been applied in an automatic speech recognition task, where the language models are essential components highly affecting the accuracy of the system. But, it is clear that these approaches are also valuable for other application areas, such as machine translation, and language generation.

In the following sections, we summarize the main contributions of this thesis.

6.1. Language Resources

We developed a set of tools and resources for efficient and effective processing of Turkish to increase the performance of SLP systems. These tools and resources are available for further studies in morphological processing of Turkish:

- We developed a stochastic finite-state morphological parser. It is the first probabilistic morphological analyzer, which outputs the parse probabilities along with the morphological analyses. It is highly efficient and can analyze about 8,700 words per second on a 2.33 GHz Intel Xeon processor.
- We developed an averaged perceptron-based morphological disambiguator, which achieves the highest disambiguation accuracy reported in the literature for Turkish. Since it is a feature-based linear model, it provides great flexibility in features that can be incorporated into the disambiguation model. The disambiguation has been implemented with a dynamic programming approach and it is fast and efficient.
- We compiled a web corpus. The compilation process and corpus statistics were

described in detail. This text corpus has been used for statistical parameter estimation throughout the thesis. This is the largest web corpus that has been used for language processing of Turkish.

6.2. Morpholexical Language Model

We proposed a novel approach for statistical language modeling of morphologically rich languages. The proposed model is called *morpholexical language model.* This model is a grammatical sub-lexical n-gram model in contrast to word or statistical sub-word models. The morpholexical n-gram language model has many advantages:

- It provides a fully linguistic approach, since the modeling units are lexical and grammatical morphemes annotated with morphosyntactic and morphosemantic features.
- The vocabulary of the model is the lexical transducer of the morphological parser, which can be considered as a computational dynamic vocabulary. Hence, the vocabulary is unlimited for Turkish.
- The out-of-vocabulary word problem is greatly alleviated with the computational lexicon. For instance, OOV rate is 1.3% on the test set, while the 200K word model has about 2% OOV rate.
- The data sparsity problem is also less pronounced since sub-lexical units are used.
- The lexicon and the morpholexical language model can be represented efficiently with finite-state transducers.
- This approach enables the use of computational pronunciation lexicons for languages where inter-morpheme coarticulation problem exists, such as Korean.
- The model generates only valid word forms when composed with a computational lexicon, hence it solves the over-generation problem of sub-lexical units.
- The lexical and morphosyntactic features can be further exploited in a rescoring or reranking model.

6.3. Morphology-Integrated Search Network

We proposed a novel approach to build a *morphology-integrated search network* for ASR with unlimited vocabulary in the weighted finite-state transducer framework (WFST). The proposed *morpholexical search network* is basically obtained by the composition of the lexical transducer of the morphological parser and the transducer of morpholexical language model. The morpholexical search network for ASR has the following advantages:

- The vocabulary of the ASR system is a dynamic vocabulary in contrast to a static fixed vocabulary with word-based models, where the vocabulary is generally dependent on the training corpus. This improves the OOV rate of the system and the recognizer can recognize the words not seen in the training corpus.
- The recognizer outputs only valid word sequences since the morpholexical language model is constrained with the morphotactics of Turkish.
- The proposed model improves ASR word error rate by 1.8% absolute over word models and 0.8% absolute over statistical sub-word models at $\sim$ 1.5 real-time factor.

6.4. Discriminative Reranking

We further improved the recognition accuracy of the morpholexical model by rescoring the n-best hypotheses with a discriminatively trained model. This model is a linear reranking model trained discriminatively with the proposed variant of the perceptron algorithm. This algorithm called WER-sensitive perceptron performs better for reranking speech recognition hypotheses by introducing error rate dependent loss function. The discriminatively trained morpholexical model improved the WER of the system by 0.8% absolute. We also showed that reranking WER for the lexical stem+ending model is lower by 2.2% and 0.7% absolute than word and morph models respectively.

6.5. Lattice Rescoring

We presented a general algorithmic framework for on-the-fly lattice rescoring. Applications such as real-time closed captioning of news broadcasts require low-latency. In such applications, offline lattice rescoring may not be used due to added latency of the rescoring, which increases with the size of the output lattice. The proposed on-the-fly lattice rescoring brings accuracy improvement of lattice rescoring without increasing the latency of the system.

As a general framework, the language model for rescoring can be any model that can assign scores to n-grams. For instance, it is possible to use the discriminative language models of this thesis to rescore the lattices in this framework. This is important since the discriminative models with complex feature sets cannot be represented as finite-state automata. The lattices can even be rescored with multiple models.

APPENDIX A: TURKISH MORPHOPHONEMICS

First we list the set symbols used in the rules to represent sets of characters from the alphabet and some special symbols:

V = {a, e, ı, i, o, ö, u, ü, â, û, î, {, }, [, %, A, E}
BV = {a, ı, o, u, â, û}
FV = {e, i, ö, ü, {, }, [, %, E, î}
CONS = {b, c, ç, d, f, g, ğ, h, j, k, l, m, n, p, q, r, s, ş, t, v, w, x, y, z}
NSY = {N, S, Y}
FSTKCSHP = {f, s, t, k, ç, ş, h, p}

The rules use the following lexical symbols to describe morphological alternations:

- The surface letters are in lower case: a, e, ı, i, o, ö, u, ü, â, û, î, b, c, ç, d, f, g, ğ, h, j, k, l, m, n, p, q, r, s, ş, t, v, w, y, z.
- + is the morpheme boundary symbol.
- ' is the apostrophe symbol used for separating some suffixes from the proper nouns.
- A is the lexical symbol realized as /a/ or /e/ in surface form.
- E is the lexical symbol realized as /e/ or /i/ in surface form. It is used to mark the exceptional alternations of the stems "de" (to say) and "ye" (to eat) as "dE" and "yE".
- H is the lexical symbol realized as /ı/, /i/, /u/, or /ü/ in surface form.
- D is the lexical symbol realized as /d/ or /t/ in surface form.
- N is the lexical symbol that can be realized as /n/ or drop in some suffixes in surface form.
- S is the lexical symbol that can be realized as /s/ or drop in some suffixes in surface form.
- Y is the lexical symbol that can be realized as /y/ or drop in some suffixes in surface form.
- " is the lexical symbol that is used to mark the stems in which degemination occurs in the final consonant.

- { is the lexical symbol realized as /a/ in surface form and behaves similar to /e/. It is used in some stems, such as “dikk{t” (attention).
- } is the lexical symbol realized as /u/ in surface form and behaves similar to /ü/. It is used in some stems, such as “us}l” (method).
- % is the lexical symbol realized as /o/ in surface form and behaves similar to /ö/. It is used in some stems, such as alk%l (alcohol).
- [is the lexical symbol that is always realized as /â/ in surface form and behaves similar to /e/.
- C is the lexical symbol realized as /ç/ or /c/ in surface form.
- G is the lexical symbol realized as /g/ in surface form.
- K is the lexical symbol realized as /k/ in surface form.
- Ç is the lexical symbol realized as /ç/ in surface form.

  These last three symbols are used to differentiate some of the exceptional stems that do not undergo default phonological alternations during suffixation. For instance, the final /k/ consonant of a stem is mostly realized as /ğ/ when a suffix starting in a vowel is affixed to and we replace it with K symbol for exceptional stems.
- ? is the lexical symbol that can be realized as /?/ when it is the single token question mark or ε when it marks the preceding vowel to drop such as “ağı?z” (mouth) in certain suffixations.
- ˜ is the lexical symbol that is always realized as ε. It is used to mark the alternation of the preceding /p/ and /t/ to /b/ and /d/, respectively, in some root words such as “kitap˜” (book).
- ˆ is the lexical symbol realized as y or ε. It is used to mark the exceptional alternations of the stems ending in “su” (water).

The two-level rules that describe Turkish phonological alternations are given below. These rules can be compiled with Xerox twolc two-level rule compiler.

```
1. A:a ⇒ [A:a | H:u | H:ı | BV:] [’:’ | :CONS | :0]* _ ;
2. A:e ⇒ [A:e | H:ü | H:i | FV:] [’:’ | :CONS | :0]* _ ;
3. H:ı ⇒ [:ı | a:a | A:a | â:â | ı:0 ?:0 | *: [ı: 0 | a:0]
```

```
         CONS: *:] (?:0) [H:0 | V:0 | CONS: | :CONS |
         ": | NSY: | ’:’ | ˜:0 | +:0]* _ ;
4. H:i ⇒ [:i | :e | i:0 ?:0 | {:a | [:â | E:0 | *:
         [i:0 | e:0 | {:0] CONS: *:] (?:0) [H:0 | V:0 |
         CONS: | :CONS | ": | NSY: | ’:’ | ˜:0 | +:0]* _ ;
5. H:u ⇒ [u:u | H:u | û:û | u:0 ?:0 | :o | *: [u:0 | o:0]
         CONS: *:] (?:0) [H:0 | V:0 | CONS: | :CONS |
         ": | NSY: | ’:’ | ˜:0 | +:0]* _ ;
6. H:ü ⇒ [:ü | ü:0 ?:0 | :ö | }:u | %:o | *: [ü:0 | ö:0]
         CONS: *:] (?:0) [H:0 | V:0 | CONS: | :CONS |
         ": | NSY: | ’:’ | ˜:0 | +:0]* _ ;
```

The first six rules implement the vowel harmony phenomenon in Turkish. The first two state that an A symbol is realized as /a/ or /e/ depending on the agreement in backness with the preceding vowel. The other four rules indicate that the realization of the H symbol is decided by the agreement in backness and roundness with the preceding vowel.

```
7. V:0 ⇔ *:0 _ (CONS:) *:0 ;
             _ +:0 H: y o r ;
             _ ?:0 :CONS (˜:0) +:0 (N:0 | S:0) [A: | H:] ;
             _ ?:0 :CONS (˜:0) +:0 Y:0 A: ;
             _ ?:0 :CONS (˜:0) +:0 Y:0 H: .#. ;
             _ ?:0 :CONS (˜:0) +:0 Y:0 H: [m:m | z:z] ;
```

This rule describes the vowel ellipsis. When a word ends in a vowel and it is attached the progressive tense suffix +Hyor, that vowel is omitted. Also, in some stems such as *ağız* (mouth), the last vowel is omitted when it is added a suffix starting with a vowel on the surface form. These stems are marked with the ? symbol after the vowel that can be omitted, such as ağı?z. The vowel omission does not occur for suffixes +YHm and +YHz.

```
8. H:0 ⇔ :V (’:’) +:0 _ ;
```

This rule is for the deletion of the H symbol when it is appended to a stem ending in surface vowel.

```
9. NSY:0 ⇔ [CONS: | :CONS] (*:0 | ^: | ~:0 (":0) | ":0 |
           ':')* +:0 _ (:CONS) :V ;
```

This rule states that the N, S, and Y symbols are deleted when they are affixed to a stem ending in consonant.

```
10. D:t ⇔ [FSTKCSHP: | :FSTKCSHP] (*:0 | ~:0 | ":0 | ':')*
          +:0 (:0) _ ;
```

The above rule says that the D symbol is realized as /t/ when it is preceded by a set of specific consonants, otherwise the D is realized as /d/ by default.

```
11. [p:b | t:d] ⇔ _ ~:0 (":) +:0 (NSY:0) [A: | H:] ;
```

This rule describes the final consonant devoicing when a suffix starting in a vowel is attached to a stem. The voiceless plosives, /p/ and /t/ symbols, are converted into their voiced counterparts /b/ and /d/, respectively, when they are at the end of a stem marked with ˜ symbol.

```
12. C:ç ⇔ :FSTKCSHP (~:0) +:0 _ [A: | H:] ;
```

This rule encodes that the C symbol in some suffixes is realized as /ç/ when they are affixed to a stem word ending in certain consonants. The default realization of this symbol is /c/.

```
13. ç:c ⇔ _ +:0 (NSY:0) :V ;
```

This rule describes the final consonant devoicing for the symbol /ç/.

```
14. k:ğ ⇔ :V _ +:0 (NSY:0) :V ;
```

This rule says that the /k/ symbol at the end of some stems is realized as /ğ/ when they are affixed a suffix starting in a vowel.

```
15. k:g ⇔ n _ +:0 (NSY:0) :V ;
```

The final /k/ consonant of a stem with a preceding /n/ consonant is converted to /g/ when a suffix starting in a vowel is affixed to.

16. `g:ğ ⇔ \[n | r] _ +:0 (NSY:0) :V ;`

The final /g/ of a stem is converted to /ğ/ when the affixed morpheme starts with a vowel, provided that the preceding consonant is not /n/ or /r/.

17. `^:y ⇔ _ +:0 [H: | NSY:0] ;`

This rule is for the exceptional case of the stem ending in “su” (water) and marked with ˆ. The ˆ symbol is replaced with the /y/ consonant when a suffix starting with a vowel is attached to.

18. `E:i ⇔ _ +:0 Y: ;`

This rule is for the stems dE (to say) and yE (to eat). The E symbol is realized as /i/ when these stems are appended a suffix starting with Y. The default realization for the E symbol is /e/.

19. `":Cx ⇔ :Cx (~:0) _ +:0 (NSY:0) [A: | H:] ; where Cx in CONS ;`

This rule is for the degemination in a small number of Arabic loan words. According to this rule, the final consonant in some stems is doubled if a suffix starting in a vowel is appended.

20. `Cx:0: ⇔ *:0 V:0 _ *:0 ; where Cx in CONS ;`

The acronyms and abbreviations in the lexicon are appended with some orthographic symbols to make their pronunciations explicit, since the morphological alternations in the affixed morphemes are dependent on their pronunciations. The added consonant symbols are deleted with this rule. The vowels are handled in the vowel omission rule.

21. `[":" | -:- | ?:? | +:+ | %:%] ⇔ .#. _ ;`

This rule is for the preservation of the punctuation symbols used also in this specification for describing some alternations. The .#. symbol indicates a word boundary.

APPENDIX B: TURKISH MORPHOTACTICS

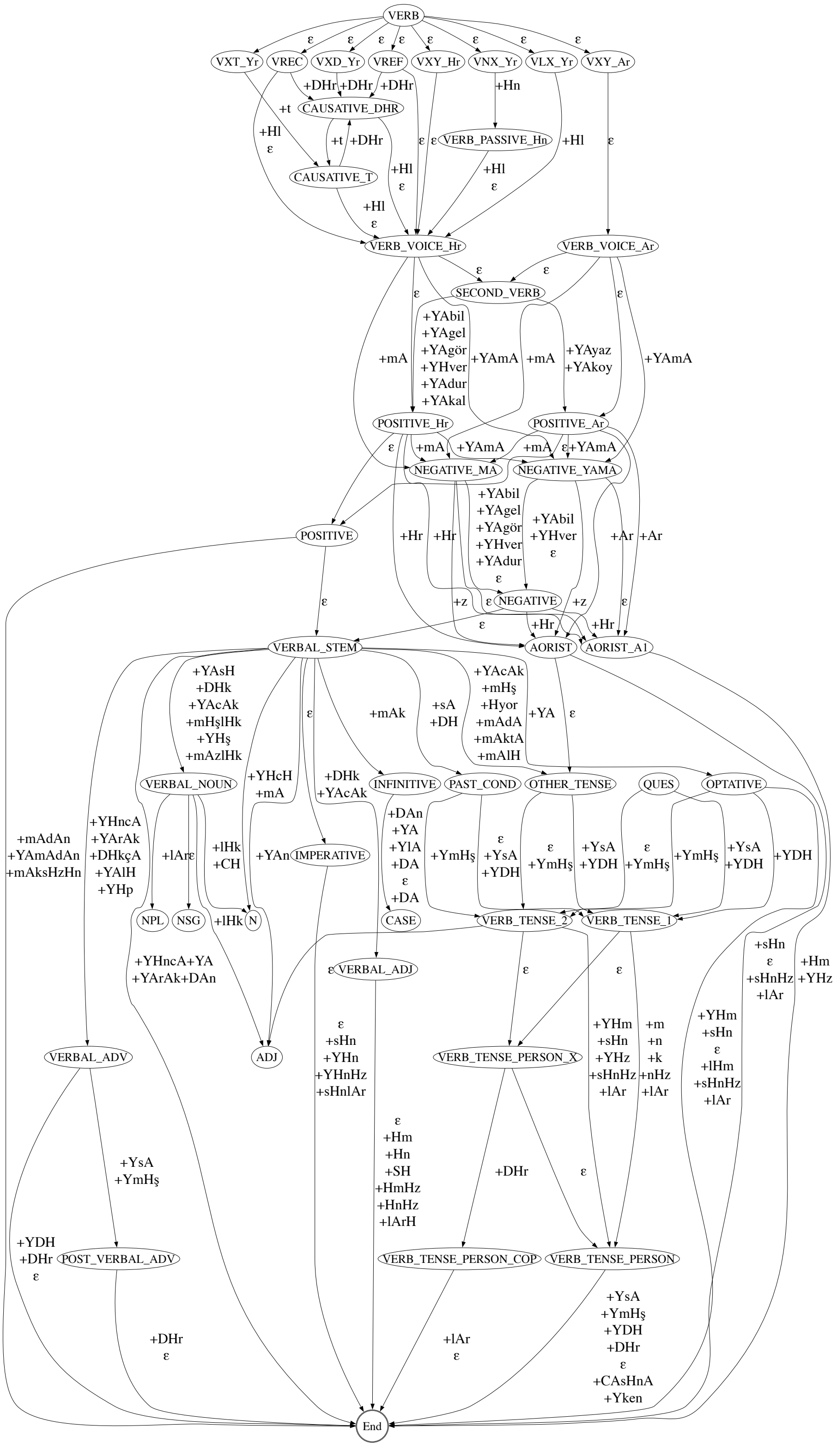


Figure B.1. Verbal Morphotactics: Morphological features are not shown for clarity.

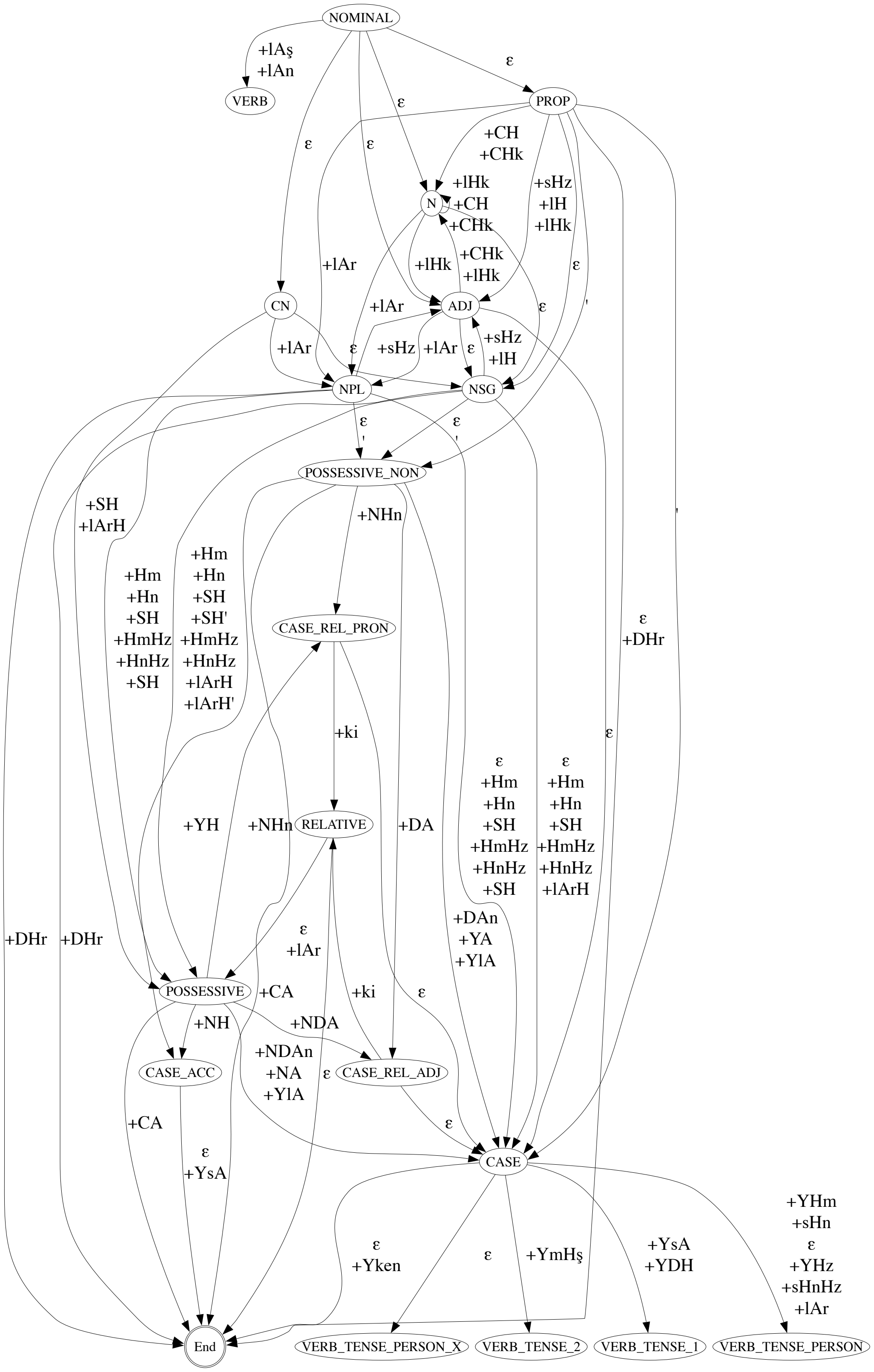


Figure B.2. Nominal Morphotactics: Morphological features are not shown for clarity.

APPENDIX C: MORPHOLOGICAL FEATURES

A list of inflectional and derivational morphological features is given here (adapted from Oflazer [94]).

Table C.1. Morphological Features.

Major Part of Speech	Gloss
+Adj	Adjective PoS
+Adv	Adverb PoS
+Conj	Conjunctive PoS
+Det	Determiner PoS
+Dup	Duplicative PoS
+Interj	Interjection PoS
+Ques	Question PoS
+Verb	Verb PoS
+PostP	Post Participle PoS
+Num	Number PoS
+Pron	Pronoun PoS
+Punc	Punctuation PoS
+Apos	Apostrophe
Minor Part of Speech	Gloss
+Card	Number Cardinal
+Ord	Number Ordinal
+Percent	Number Percentage
+Range	Number Range
+Real	Number Real
+Ratio	Number Ratio
+Distrib	Number Distribution
+Time	Number Time
+Inf	Noun Infinitival
+PastPart	Noun Past Participle
+FutPart	Noun Future Participle
+Prop	Noun Proper
+PastPart	Adjective Past Participle
+FutPart	Adjective Future Participle
+PresPart	Adjective Present Participle
+DemonsP	Pronoun Demonstrative
+QuesP	Pronoun Question
+ReflexP	Pronoun Reflexive
+PersP	Pronoun Person
+QuantP	Pronoun Quantitative

Table C.2. Morphological Features. (cont.)

Number/Person Agreement	Gloss
+A1sg	First Person Singular
+A2sg	Second Person Singular
+A3sg	Third Person Singular
+A1pl	First Person Plural
+A2pl	Second Person Plural
+A3pl	Third Person Plural
Possessive Agreement	Gloss
+Pnon	No possessive marker
+P1sg	First Person Singular
+P2sg	Second Person Singular
+P3sg	Third Person Singular
+P1pl	First Person Plural
+P2pl	Second Person Plural
+P3pl	Third Person Plural
Nominal Case	Gloss
+Abl	Ablative
+Acc	Accusative
+Dat	Dative
+Equ	Equative
+Gen	Genitive
+Ins	Instrumental
+Loc	Locative
+Nom	Nominative

Table C.3. Morphological Features. (cont.)

Verb Derivational Markers	Gloss
+Pass	Passive
+Caus	Causative
+Reflex	Reflexive
+Recip	Reciprocal
+Able	able to Verb
+Repeat	Verb repeatedly
+Hastily	Verb hastily
+EverSince	have been verb-ing ever since
+Almost	almost verb-ed
+Stay	stayed frozen while verb-ing
+Start	start verb-ing immediately
Verb Inflectional Markers	Gloss
+Pos	Positive Polarity
+Neg	Negative Polarity
+Past	Past Tense
+Narr	Narrative Past Tense
+Fut	Future Tense
+Aor	Aorist
+Pres	Present Tense
+Desr	Desire/Wish
+Cond	Conditional
+Neces	Necessitative
+Opt	Optative
+Imp	Imperative
+Cop	Copula
+Prog1	Present Continuous Process
+Prog2	Present Continuous State
Semantic Markers	
+AfterDoingSo +SinceDoingSo +As +When +ByDoingSo +While +AsIf +WithoutHavingDoneSo +Since +With +Without +SuitableFor +InBetween +Relative +Ness +Agt +Dim +Become +Acquire	

APPENDIX D: PROOF OF THE CONVERGENCE OF THE WER-SENSITIVE PERCEPTRON

We give a proof of the convergence of the WER-sensitive perceptron algorithm for a separable training sequence following a similar proof given for the averaged perceptron in [1].

Definition D.1. *Let the set of incorrect candidates for an example x_i be $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \mathcal{Y}_i$ where $\mathcal{Y}_i$ is the set of all candidates with the lowest error rate. The training sequence $(x_i, y_i \in \mathcal{Y}_i)$ for $i = 1 \ldots n$ is separable with margin $\delta > 0$ if there exists some vector $\mathbf{U}$ with $||\mathbf{U}|| = 1$ such that*

$$\forall i, \forall z_i \in \overline{\mathbf{GEN}}(x_i), \mathbf{U} \cdot \mathbf{\Phi}(x_i, y_i) - \mathbf{U} \cdot \mathbf{\Phi}(x_i, z_i) \geq \delta \tag{D.1}$$

Theorem D.1. *For any training sequence (x_i, y_i) which is separable with margin δ, for the perceptron algorithm in Figure 4.1:*

$$\textit{Number of mistakes} \leq \frac{R^2 r^2}{\delta^2} \tag{D.2}$$

where R is a constant such that $\forall i, \forall z_i \in \overline{\mathbf{GEN}}(x_i)$, $||\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)|| \leq R$ and r is an upper bound on loss for any candidate, that is $\forall i, \forall z_i \in \overline{\mathbf{GEN}}(x_i)$, $\Delta_i(z_i, y_i) \leq r$ where $\Delta_i(z_i, y_i)$ is the difference in the edit distances of z_i and y_i with the reference transcription of x_i, and $y_i \in \mathcal{Y}_i$ is a candidate with the lowest error rate.

Proof. Suppose that k^{th} mistake is made at the i^{th} example and let $\bar{\alpha}^k$ be the weights before that mistake is made and hence $\bar{\alpha}^1 = 0$. Take z_i as the output proposed at this example, $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \bar{\alpha}^k$. It follows from the algorithm updates that $\bar{\alpha}^{k+1} = \bar{\alpha}^k + (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i))\Delta_i(z_i, y_i)$. First we derive a lower bound for

$||\bar{\alpha}^{k+1}||$:

$$\begin{aligned}\mathbf{U}\cdot\bar{\alpha}^{k+1} &= \mathbf{U}\cdot\bar{\alpha}^{k}+\mathbf{U}\cdot(\mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_i,z_i))\Delta_i(z_i,y_i)\\ &\geq \mathbf{U}\cdot\bar{\alpha}^{k}+\delta\Delta_i(z_i,y_i)\end{aligned}$$

where the inequality follows from the definition of $\mathbf{U}$. Since $\mathbf{U}\cdot\bar{\alpha}^{1}=0$, it follows by induction on k that for all k, $\mathbf{U}\cdot\bar{\alpha}^{k+1}\geq\delta\sum_k\Delta_i(z_i,y_i)$ where $\sum_k\Delta_i(z_i,y_i)$ is the sum of losses made at each mistake up to k^{th} mistake. Because $\mathbf{U}\cdot\bar{\alpha}^{k+1}\leq||\mathbf{U}||||\bar{\alpha}^{k+1}||$ and $||\mathbf{U}||=1$, it follows that $||\bar{\alpha}^{k+1}||\geq\delta\sum_k\Delta_i(z_i,y_i)$. Because the loss is at least one for each mistake by definition, it follows that $||\bar{\alpha}^{k+1}||\geq\delta k$

Now we derive an upper bound for $||\bar{\alpha}^{k+1}||^2$:

$$\begin{aligned}||\bar{\alpha}^{k+1}||^2 &= ||\bar{\alpha}^{k}||^2+||\mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_i,z_i)||^2\Delta_i(z_i,y_i)^2+\\ &\quad 2\bar{\alpha}^{k}\cdot(\mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_i,z_i))\Delta_i(z_i,y_i)\\ &\leq ||\bar{\alpha}^{k}||^2+R^2\Delta_i(z_i,y_i)^2\end{aligned}$$

where the inequality follows because $||\mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_i,z_i)||^2\leq R^2$ by assumption, and $\bar{\alpha}^{k}\cdot(\mathbf{\Phi}(x_i,y_i)-\mathbf{\Phi}(x_i,z_i))\leq 0$ because z_i is the highest scoring candidate for x_i under the parameters $\bar{\alpha}^{k}$. It follows by induction that $||\bar{\alpha}^{k+1}||^2\leq R^2\sum_k\Delta_i(z_i,y_i)^2$. Because the loss for any candidate has an upper bound r by assumption, it follows that $||\bar{\alpha}^{k+1}||^2\leq R^2kr^2$.

Combining the bounds $||\bar{\alpha}^{k+1}||\geq\delta k$ and $||\bar{\alpha}^{k+1}||^2\leq R^2kr^2$ gives the result for all k that

$$\delta^2k^2\leq||\bar{\alpha}^{k+1}||^2\leq R^2kr^2\Rightarrow k\leq\frac{R^2r^2}{\delta^2}.$$

Because the upper bound on the number of mistakes the algorithm makes is constant, the algorithm must converge within a finite number of iterations. □

REFERENCES

1. Collins, M., "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–8, 2002.

2. Ljolje, A., F. Pereira and M. Riley, "Efficient General Lattice Generation and Rescoring", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1251–1254, 1999.

3. Arısoy, E., *Statistical and Discriminative Language Modeling for Turkish Large Vocabulary Continuous Speech Recognition*, Ph.D. Thesis, Boğaziçi University, 2009.

4. Allauzen, C., M. Mohri and B. Roark, "Generalized Algorithms for Constructing Statistical Language Models", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 40–47, 2003.

5. Mohri, M., "Finite-State Transducers in Language and Speech Processing", *Computational Linguistics*, Vol. 23, No. 2, pp. 269–311, 1997.

6. Mohri, M., F. Pereira and M. Riley, "Weighted Finite-State Transducers in Speech Recognition", *Computer Speech and Language*, Vol. 16, No. 1, pp. 69–88, 2002.

7. Mohri, M. and M. Riley, "Integrated Context-Dependent Networks in Very Large Vocabulary Speech Recognition", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pp. 811–814, 1999.

8. Allauzen, C., M. Mohri, M. Riley and B. Roark, "A Generalized Construction of Integrated Speech Recognition Transducers", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 761–764, 2004.

9. Salomaa, A. and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag, New York, 1978.

10. Berstel, J. and L. Boasson, *Transductions and Context-Free Languages*, Teubner Verlag, Leipzig, 1979.

11. Berstel, J. and C. Reutenauer, *Rational Series and their Languages*, Springer-Verlag, New York, 1988.

12. Eilenberg, S., *Automata, Languages, and Machines*, Academic Press, New York, 1974.

13. Kuich, W. and A. Salomaa, *Semirings, Automata, Languages*, Springer-Verlag, New York, 1986.

14. Mohri, M., *Handbook of Weighted Automata. Monographs in Theoretical Computer Science*, chap. Weighted Automata Algorithms, pp. 213–254, Springer-Verlag, Berlin, 2009.

15. Mohri, M., *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition.*, chap. Speech Recognition with Weighted Finite-State Transducers, Springer-Verlag, Berlin, 2008.

16. Collins, M. and N. Duffy, "New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 263–270, 2002.

17. Collins, M. and T. Koo, "Discriminative Reranking for Natural Language Parsing", *Computational Linguistics*, Vol. 31, No. 1, pp. 25–70, 2005.

18. Shen, L. and A. K. Joshi, "Ranking and Reranking with Perceptron", *Machine Learning*, Vol. 60, No. 3, pp. 73–96, 2005.

19. Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage

and Organization in the Brain", *Psychological Review*, Vol. 65, No. 6, pp. 386–408, 1958.

20. Freund, Y. and R. E. Schapire, "Large Margin Classification Using the Perceptron Algorithm", *Machine Learning*, Vol. 37, No. 3, pp. 277–296, 1999.

21. Rosenfeld, R., *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. Thesis, Carnegie Mellon University, 1994.

22. Singh-Miller, N. and M. Collins, "Trigger-Based Language Modeling Using a Loss-Sensitive Perceptron Algorithm", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 4, pp. 25–28, 2007.

23. Jelinek, F., *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, Massachusetts, 1998.

24. Jurafsky, D. and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*, Prentice Hall, New Jersey, 2000.

25. Rabiner, L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.

26. Baum, L. E., T. Petrie, G. Soules and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains", *The Annals of Mathematical Statistics*, Vol. 41, No. 1, pp. 164–171, 1970.

27. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1, pp. 1–38, 1977.

28. Viterbi, A., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory*, Vol. 13,

No. 2, pp. 260–269, 1967.

29. Saon, G., D. Povey and G. Zweig, "Anatomy of an Extremely Fast LVCSR Decoder", *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 549–552, 2005.

30. Stolcke, A., "Entropy-based Pruning of Backoff Language Models", *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 270–274, 1998.

31. Kneser, R. and H. Ney, "Improved Backing-off for M-gram Language Modeling", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 181–184, 1995.

32. Chen, S. F. and J. Goodman, "An Empirical Study of Smoothing Techniques for Language Modeling", *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pp. 310–318, Association for Computational Linguistics, Stroudsburg, PA, USA, 1996.

33. Rosenfeld, R., "Two Decades of Statistical Language Modeling: Where Do We Go from Here?", *Proceedings of the IEEE*, Vol. 88, No. 8, pp. 1270–1278, 2000.

34. Roark, B., M. Saraçlar and M. Collins, "Discriminative N-gram Language Modeling", *Computer Speech and Language*, Vol. 21, No. 2, pp. 373–392, 2007.

35. Hetherington, I. L., *A Characterization of the Problem of New, Out-of-Vocabulary Words in Continuous-Speech Recognition and Understanding*, Ph.D. Thesis, Massachusetts Institute of Technology, 1995.

36. Rosenfeld, R., "Optimizing Lexical and N-gram Coverage Via Judicious Use of Linguistic Data", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1763–1766, 1995.

37. Oflazer, K. and S. Inkelas, "The Architecture and the Implementation of a Finite

state Pronunciation Lexicon for Turkish", *Computer Speech and Language*, Vol. 20, pp. 80–106, 2006.

38. Arısoy, E., D. Can, S. Parlak, H. Sak and M. Saraçlar, "Turkish Broadcast News Transcription and Retrieval", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 17, No. 5, pp. 874–883, 2009.

39. Stolcke, A., "SRILM – an Extensible Language Modeling Toolkit", *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Vol. 2, pp. 901–904, 2002.

40. Geutner, P., "Using Morphology Towards Better Large-Vocabulary Speech Recognition Systems", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 445–448, 1995.

41. Ircing, P., P. Krbec, J. Hajic, J. Psutka, S. Khudanpur, F. Jelinek and W. Byrne, "On Large Vocabulary Continuous Speech Recognition of Highly Inflectional Language - Czech", *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 487–490, 2001.

42. Kwon, O.-W. and J. Park, "Korean Large Vocabulary Continuous Speech Recognition with Morpheme-Based Recognition Units", *Speech Communication*, Vol. 39, No. 3-4, pp. 287 – 300, 2003.

43. Choueiter, G., D. Povey, S. F. Chen and G. Zweig, "Morpheme-Based Language Modeling for Arabic LVCSR", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 1, pp. 1053–1056, 2006.

44. Rotovnik, T., M. S. Maučec and Z. Kačič, "Large Vocabulary Continuous Speech Recognition of an Inflected Language Using Stems and Endings", *Speech Communication*, Vol. 49, pp. 437–452, 2007.

45. Goldsmith, J., "Unsupervised Learning of the Morphology of a Natural Language",

Computational Linguistics, Vol. 27, No. 2, pp. 153–198, 2001.

46. Creutz, M. and K. Lagus, "Unsupervised Discovery of Morphemes", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 21–30, 2002.

47. Siivola, V., T. Hirsimäki, M. Creutz and M. Kurimo, "Unlimited Vocabulary Speech Recognition Based on Morphs Discovered in an Unsupervised Manner", *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 2293–2296, 2003.

48. Hirsimäki, T., M. Creutz, V. Siivola, M. Kurimo, S. Virpioja and J. Pylkkönen, "Unlimited Vocabulary Speech Recognition with Morph Language Models Applied to Finnish", *Computer Speech and Language*, Vol. 20, No. 4, pp. 515–541, 2006.

49. Çarkı, K., P. Geutner and T. Schultz, "Turkish LVCSR: Towards Better Speech Recognition For Agglutinative Languages", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 3, pp. 1563–1566, 2000.

50. Hakkani-Tür, D. Z., *Statistical Language Modeling for Agglutinative Languages*, Ph.D. Thesis, Bilkent University, 2000.

51. Dutağacı, H., *Statistical Language Models for Large Vocabulary Continuous Speech Recognition of Turkish*, M.S. Thesis, Boğaziçi University, 2002.

52. Arısoy, E., *Turkish Dictation System for Radiology and Broadcast News Applications*, M.S. Thesis, Boğaziçi University, 2004.

53. Hacıoğlu, K., B. L. Pellom, T. Çiloğlu, O. Öztürk, M. Kurimo and M. Creutz, "On Lexicon Creation for Turkish LVCSR", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1165–1168, 2003.

54. Erdoğan, H., O. Büyük and K. Oflazer, "Incorporating Language Constraints in

Sub-Word Based Speech Recognition", *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 98–103, 2005.

55. Arısoy, E. and M. Saraçlar, "Lattice extension and vocabulary adaptation for Turkish LVCSR", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 17, No. 1, pp. 163–173, 2009.

56. Kirchhoff, K., D. Vergyri, J. Bilmes, K. Duh and A. Stolcke, "Morphology-Based Language Modeling for Conversational Arabic Speech Recognition", *Computer Speech and Language*, Vol. 20, No. 4, pp. 589 – 608, 2006.

57. Afify, M., R. Sarikaya, H.-K. J. Kuo, L. Besacier and Y. Gao, "On the Use of Morphological Analysis for Dialectal Arabic Speech Recognition", *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 277–280, 2006.

58. Sarikaya, R., M. Afify, Y. Deng, H. Erdogan and Y. Gao, "Joint Morphological-Lexical Language Modeling for Processing Morphologically Rich Languages With Application to Dialectal Arabic", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 16, No. 7, pp. 1330–1339, 2008.

59. Lewis, G., *Turkish Grammar*, Oxford University Press, New York, 2001.

60. Oflazer, K., "Two-level Description of Turkish Morphology", *Literary and Linguistic Computing*, Vol. 9, No. 2, pp. 137–148, 1994.

61. Antworth, E. L., *PC-KIMMO: A Two-Level Processor for Morphological Analysis*, Lawrence Erlbaum Associates, New York, 1990.

62. Karttunen, L. and K. R. Beesley, *Two-level Rule Compiler*, Technical report, Xerox Palo Alto Research Center, Palo Alto, CA, 1992.

63. Öztaner, S. M., *A Word Grammar of Turkish with Morphophonemic Rules*, M.S. Thesis, Middle East Technical University, 1996.

64. Güngör, T., *Computer Processing of Turkish: Morphological and Lexical Investigation*, Ph.D. Thesis, Boğaziçi University, 1995.

65. Hajic, J. and B. Hladká, "Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset", *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pp. 483–490, 1998.

66. Ezeiza, N., I. Alegria, J. M. Arriola, R. Urizar and I. Aduriz, "Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages", *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pp. 380–384, 1998.

67. Megyesi, B., "Improving Brill's PoS Tagger for an Agglutinative Language", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 275–284, 1999.

68. Oflazer, K. and G. Tür, "Combining Hand-Crafted Rules and Unsupervised Learning in Constraint-Based Morphological Disambiguation", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 69–81, Somerset, New Jersey, 1996.

69. Oflazer, K. and G. Tür, "Morphological Disambiguation by Voting Constraints", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 222–229, 1997.

70. Hakkani-Tür, D. Z., K. Oflazer and G. Tür, "Statistical Morphological Disambiguation for Agglutinative Languages", *Computers and the Humanities*, Vol. 36, No. 4, pp. 285–291, 2002.

71. Yüret, D. and F. Türe, "Learning Morphological Disambiguation Rules for Turkish", *HLT-NAACL*, pp. 328–334, 2006.

72. Sak, H., T. Güngör and M. Saraçlar, "Morphological Disambiguation of Turkish Text with Perceptron Algorithm", *CICLing 2007*, Vol. LNCS 4394, pp. 107–118, 2007.

73. Say, B., D. Zeyrek, K. Oflazer and U. Özge, "Development of a Corpus and a Treebank for Present-day Written Turkish", *Proceedings of the Eleventh International Conference of Turkish Linguistics*, pp. 183–192, 2002.

74. Salor, Ö., B. L. Pellom, T. Ciloglu, K. Hacioglu and M. Demirekler, "On Developing New Text and Audio Corpora and Speech Recognition Tools for the Turkish Language", *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pp. 349–352, 2002.

75. Allauzen, C., M. Riley, J. Schalkwyk, W. Skut and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-State Transducer Library", *CIAA 2007*, Vol. 4783 of *LNCS*, pp. 11–23, Springer, 2007, `http://www.openfst.org`.

76. Koskenniemi, K., "A General Computational Model for Word-Form Recognition and Production", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 178–181, 1984.

77. Oflazer, K., B. Say, D. Z. Hakkani-Tür and G. Tür, *Building and Exploiting Syntactically-Annotated Corpora*, chap. Building a Turkish Treebank, Kluwer Academic Publishers, 2003.

78. Kaplan, R. M. and M. Kay, "Regular models of phonological rule systems", *Computational Linguistics*, Vol. 20, pp. 331–378, 1994.

79. Karttunen, L., K. Koskenniemi and R. M. Kaplan, *A Compiler for Two-level Phonological Rules*, Technical report, Center for the Study of Language and Information, Stanford University, Palo Alto, CA, 1987.

80. Karttunen, L., R. M. Kaplan and A. Zaenen, "Two-Level Morphology with Compo-

sition", *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 141–148, 1992.

81. Kilgarriff, A. and G. Grefenstette, "Introduction to the Special Issue on the Web as Corpus", *Computational Linguistics*, Vol. 29, No. 3, pp. 333–348, 2003.

82. Liu, V. and J. R. Curran, "Web Text Corpus for Natural Language Processing", *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 233–240, Italy, 2006.

83. Eisner, J., "Parameter Estimation for Probabilistic Finite-State Transducers", *Proceedings of the Association for Computational Linguistics (ACL) Conference*, pp. 1–8, 2002.

84. Solak, A. and K. Oflazer, "Design and Implementation of a Spelling Checker for Turkish", *Literary and Linguistic Computing*, Vol. 8, No. 3, pp. 113–130, 1993.

85. Oflazer, K., "Error-Tolerant Finite-State Recognition with Applications to Morphological Analysis and Spelling Correction", *Computational Linguistics*, Vol. 22, No. 1, pp. 73–89, 1996.

86. Sak, H., M. Saraçlar and T. Güngör, "Morphology-Based and Sub-word Language Modeling for Turkish Speech Recognition", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5402–5405, 2010.

87. Arısoy, E., H. Sak and M. Saraçlar, "Language Modeling for Automatic Turkish Broadcast News Transcription", *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 2381–2384, 2007.

88. Allauzen, C., M. Mohri and B. Roark, "The Design Principles and Algorithms of a Weighted Grammar Library", *International Journal of Foundations of Computer Science*, Vol. 16, No. 3, pp. 403–421, 2005.

89. Arısoy, E., M. Saraçlar, B. Roark and I. Shafran, “Discriminative Language Modeling with Linguistic and Statistically Derived Features”, *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 20, No. 2, pp. 540–550, 2011.

90. Saraçlar, M., M. Riley, E. Bocchieri and V. Goffin, “Towards Automatic Closed Captioning: Low Latency Real Time Broadcast News Transcription”, *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pp. 1741–1744, 2002.

91. Kuo, H.-K., B. Kingsbury and G. Zweig, “Discriminative Training of Decoding Graphs for Large Vocabulary Continuous Speech Recognition”, *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 4, pp. 45–48, 2007.

92. Hori, T., C. Hori, Y. Minami and A. Nakamura, “Efficient WFST-Based One-Pass Decoding With On-The-Fly Hypothesis Rescoring in Extremely Large Vocabulary Continuous Speech Recognition”, *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 15, No. 4, pp. 1352–1365, 2007.

93. Lehr, M. and I. Shafran, “Discriminatively Estimated Joint Acoustic, Duration and Language Model for Speech Recognition”, *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5542–5545, 2010.

94. Kemal Oflazer, D. Z. H.-T. and G. Tür, “Design for a Turkish Treebank”, *In Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL’99)*, pp. 99–104, 1999.