COMPUTER-AIDED TRANSCRIPTION TOOL

by

Çağdaş Kayra Akman

B.S., in E.E., Boğaziçi University, 2004

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

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2007

COMPUTER-AIDED TRANSCRIPTION TOOL

APPROVED BY:

Assist. Prof. Murat Saraçlar (Thesis Supervisor)	.....
Prof. Levent M. Arslan	.....
Prof. Fikret Gürgen	.....

DATE OF APPROVAL: 25.01.2007

ACKNOWLEDGEMENTS

I would like to thank my adviser Murat Saraçlar for his guidance, patience and tolerance throughout the preparation of this thesis. He supported me most when he had all the reasons to be disappointed in me. He made me realize what it is like to be a researcher, to think like a, as a, researcher. He has given me much more than I could reflect in this thesis. I am very grateful to him.

I would also like to thank Levent Arslan not, only, for being in my thesis committee but for being the person who introduced me to speech and language processing area.

Prof. Fikret Gürgen has very kindly accepted to be in my thesis committee. I am grateful to him.

I owe a great debt of gratitute to Ebru. Among all those things for which she couldn't find enough time to do, she has always come to my rescue when I needed. She not only provided the test and traininig data used in this thesis, but also helped me learn so many things, from HMMs to shell scripting, I couldn't have learnt by myself in such a short time.

I don't know how I can possibly express my gratitute to my family. Without them, I wouldn't be where I am know and where I will be. They have always been the most important factor in everything I have accomplished in my life.

This work was funded partly through a scholarship granted to the author by TÜBİTAK - BİDEB and partly by projects funded by Research Fund of Boğaziçi University (Project code: 05HA202) and TÜBİTAK (Project code: 105E102).

ABSTRACT

COMPUTER-AIDED TRANSCRIPTION TOOL

State-of-the-art speech recognition and language processing systems widely use data-driven methods. These methods require large transcribed speech and annotated text corpora. The success of these systems greatly depends on the amount of the training data. Need for transcribed speech makes transcription an important component of every system employing statistical methods. Manual transcription is an expensive and slow task. Computers may do the same task much faster but with more errors. Computer Aided Transcription is a combination between these two methods. The output lattices of an ASR engine, which contain hypotheses about the utterances to be transcribed, are transformed into letter-based, deterministic, weighted finite-state acceptors. These transformed lattices are combined with a letter-based N-gram language model trained on a text corpus similar in content to the speech data. The combined model is used as the language model of the open source graphical text entry application Dasher, developed at the University of Cambridge. Lattice expansion methods are used to increase the performance of the combined model. It is shown that combining the models at letter level performs better than a letter-based N-gram model used as the only language model and the model built by combining the transformed lattices and letter-based N-gram model at sentence level.

ÖZET

BİLGİSAYAR DESTEKLİ ÇEVRİYAZI ARACI

Konuşma tanıma ve dil işleme sistemlerinde yaygın olarak veriye dayalı yöntemler kullanılmaktadır. Bu yöntemler büyük boyutlu yazıya çevrilmiş konuşma ve işlenmiş metin derlemleri gerektirmektedirler. Bu dizgelerin başarılı olmaları büyük ölçüde eğitim verisinin miktarına bağlıdır. Yazıya çevrilmiş konuşma gereksinimi, çevriyazmayı istatistiksel yöntemler kullanan her dizgenin önemli bir bileşeni yapmaktadır. Elle çevriyazım pahalı ve yavaş bir işlemdir. Bilgisayarlar aynı görevi daha hızlı ama daha çok hata yaparak gerçekleştirebilirler. Bilgisayar Destekli Çevriyazı bu iki yöntemin birleştirilmesidir. Çevriyazılacak konuşmalarla ilgili hipotezleri içeren ve bir konuşma tanıma motorunun çıktısı olan örüler, harf tabanlı, gerekirci, ağırlık sonlu durum alıcılarına dönüştürülmüştür. Bu dönüştürülmüş örüler içerik bakımından konuşma verisiyle örtüşen bir metin derlemiyle eğitilmiş harf tabanlı istatistiksel bir dil modeliyle birleştirilmiştir. Birleşik model Cambridge Üniversitesi'nde geliştirilmekte olan açık kaynak kodlu bir metin girişi uygulaması olan Dasher'ın dil modeli olarak kullanılmıştır. Birleşik modelin başarımını artırmak için örü genişletme yöntemleri kullanılmıştır. Modelleri harf düzeyinde birleştirmenin, tek model olarak kullanılan harf tabanlı bir istatistiksel dil modelinden ve modellerin cümle düzeyinde birleştirilmesiyle oluşturulan modelden daha iyi sonuçlar verdiği gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\mathbf{A}$	Acoustic data vector
$b(h_i)$	The node n_k in the lattice such that $n_k \in h_i$, $s(n_k) = \#$ and $\# \notin c_k^i$
c_i	Character i on an arc going out of n_i
c_k^l	The character sequence $c_k c_{k+1} \cdots c_{l-1} c_l$
$C(A(n_i))$	Set of characters that are on arcs leaving nodes in $A(n_i)$ where A is one of N, W or Φ
$C(n_i)$	Short notation for $C(\{n_i\})$
$CNT(w_{i-N+1}^i)$	The count of the N-gram context w_{i-N+1}^i
$f(h_i)$	n_i
h_i	$n_1 c_1 n_2 c_2 \cdots n_{i-1} c_{i-1} n_i$: node-character sequence leading to n_i
$l(h_i)$	The last node in h_i that is present in the lattice
n_i	Node i in the lattice or N-gram model
$n(c_i)$	n_i
$N(n_i)$	Set of nodes in the lattice accessible from n_i by skipping at least one character up to and including #
p	Geometric distribution parameter
$P_L(a)$	Probability of a computed from the lattice
$P_N(a)$	Probability of a computed from the N-gram model
q	Geometric distribution parameter
$s(n_i)$	c_{i-1}
w_i	The ith word in a word sequence
w_k^l	The word sequence $w_k w_{k+1} \cdots w_{l-1} w_l$
$\mathbf{W}$	Word sequence
$\hat{\mathbf{W}}$	Most probable word sequence
$W(n_i)$	$\Phi(n_i) \bigcup N(n_i)$
α	Normalization coefficient in backoff and interpolation models
γ	A parameter in backoff and interpolation models

λ	A parameter in backoff and interpolation models
$\Phi(n_i)$	$\{n_i\} \bigcup$ (Set of nodes in the lattice accessible from n_i without consuming any characters)
#	Word boundary symbol
ASR	Automatic Speech Recognition
BO	Backoff
CATT	Computer-Aided Transcription Tool
CE	Cross Entropy
char.	Character
DI	Deleted Interpolation
HMM	Hidden Markov Model
FSA	Finite-State Acceptor
FSM	Finite-State Machine
FST	Finite-State Transducer
GUI	Graphical User Interface
IP	Interpolation
KN	Kneser-Ney
LLC	Letter Level Combination
LM	Language Model
OOV	Out of Vocabulary
PPM	Prediction by Partial Match
R&D	Research and Development
SLC	Sentence Level Combination
SD	Speech Dasher
WER	Word Error Rate
WFSA	Weighted Finite-State Acceptor
WFSM	Weighted Finite-State Machine
WFST	Weighted Finite-State Transducer
wrt	with respect to

1. INTRODUCTION

Statistical methods are popular in speech and language processing research. Statistical inference requires processing of large amounts of speech or text data. More transcribed speech data is always needed. This is more so for languages other than English because such corpora are either not available or insufficient in amount.

Transcription is a labor-intensive and error-prone task. Manual transcription is not only a slow process but also a costly one. Moreover, errors are inevitable and difficult to correct. An alternative to manual transcription may be ASR-based transcription. Despite the success of recently introduced lightly supervised techniques, it is impractical to rely only on an ASR-based transcription when the error rates are unacceptably high.

Computer aided transcription is a compromise between manual and machine transcription. It utilizes the recognition results of an ASR engine and presents it to the transcriber for fast and efficient editing. A computer tool for such a task must have two main components: a graphical user interface (GUI) that facilitates text entry and editing by the transcriber, and a mechanism that processes the output of an ASR engine and makes use of the information it provides, even in the case of poor recognition. Part of this mechanism is a language model that combines the ASR output with a basic language model. While the ASR output is emphasized for easier transcription, the basic language model covers character sequences not present in the ASR output.

The motivation for this thesis lies in the need for an efficient transcription application that may be used for an ongoing project in the BUSIM Laboratory of Boğaziçi University. The aim of the project is to develop a speech recognition and information retrieval system for Turkish broadcast news.

1.1. Data Entry and Computer Interfaces

A conventional computer keyboard is the main means of text entry into computers. It has a static key layout to which a user gets accustomed over time. Although someone can become quite fast using a keyboard, it has some shortcomings in terms of usability.

Keyboard usage requires relatively greater physical effort than other text entry interfaces such as mouse, touchpad or stylus. It requires both hands to be used. Keyboards are memoryless devices, they don't make use of the regularities of the language being used.

Research on computer interfaces for text entry focuses not only to desktop computers but all sorts of electronic devices, such as cellular phones, PDAs, etc., that provide an interface for the users to communicate information to computers [1]. However, since the subject of this thesis is computer-aided transcription, the discussion is confined to conventional computers and their standard peripheral device for data entry, the keyboard.

1.1.1. Dasher

The open source Dasher application, which is developed in the University of Cambridge, is a graphical data entry interface operated by continuous gestures, such as mouse, touch screen, or eye-tracker [2]. Users zoom in on letters presented in different sizes on the display depending on the characters entered so far. The more probable characters are given a greater share of the display area to facilitate the selection of those characters. Moving the pointer towards the desired character suffices for its selection, as seen in Figure 1.1 [3]. No mouse clicks or key strokes are required. Dasher can be operated using only one hand. The entered text can be saved into a text file after finishing text entry.

Dasher uses a language model based on the prediction by partial match algorithm

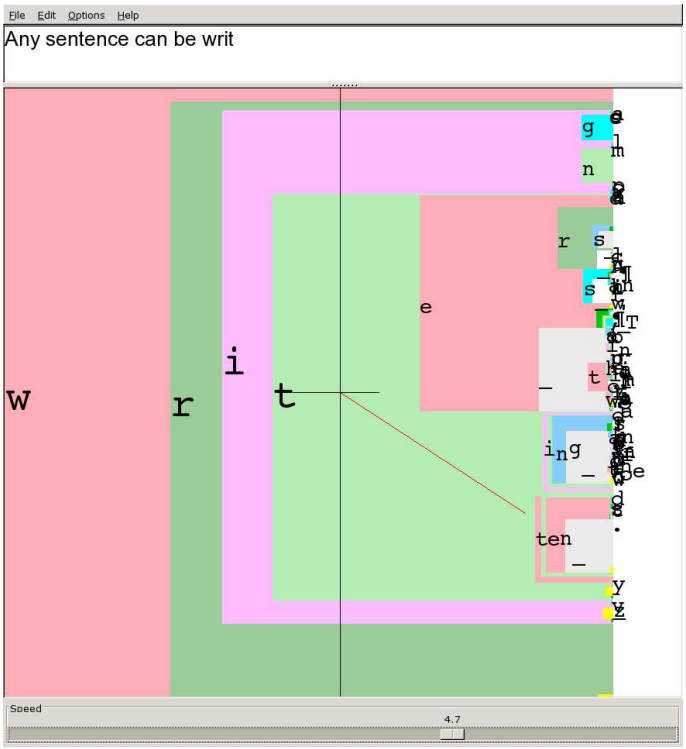

Figure 1.1. A Screenshot of Dasher

(PPM) [1]. Similar to N-gram modeling, PPM considers a finite length context and generates a probability distribution on the fly [4]. The user is presented with a continuously modified screen layout of characters differing in size according to the underlying language model. This lowers the probability that a wrong character is entered. Dasher provides the GUI part of the computer aided transcription tool (CATT).

1.1.2. PPM

PPM is a text compression algorithm that uses finite-context models of characters [5]. Since the length of the context used depends on the previous contexts observed, it is called prediction by partial matching. PPM encodes a given character according to the current probability distribution that is generated based on the text seen so far. Since the decoder updates the probability distribution with decoded text as well, correct characters are assigned to the code to be decoded. The principal idea underlying the PPM method is the same as N-gram language modeling. Context of the current

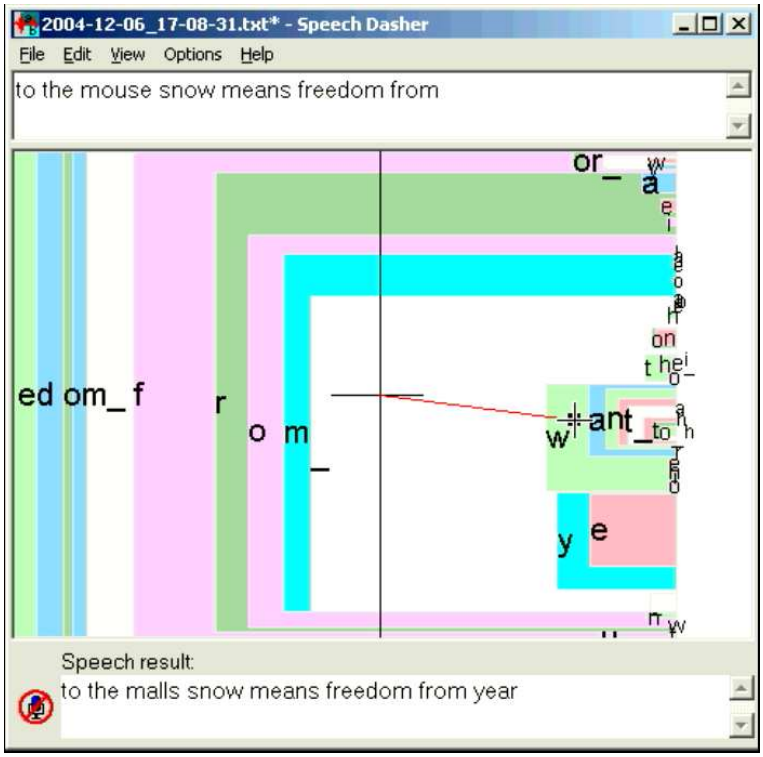

Figure 1.2. A Screenshot of Speech Dasher

symbol is used to predict the probability of occurrence of the that symbol given the context. There are a variety of the so called *escaping* schemes which correspond to *smoothing* and *backoff* to lower order contexts in N-gram modeling [5]. In fact, the smoothing method known as Witten-Bell smoothing was originally called the Method C of PPM modeling in [6].

1.1.3. Speech Dasher

Speech Dasher is an application based on Dasher that is developed to facilitate correction of speech recognition errors. It adds closed source speech recognition engine of Microsof to Dasher and aims to form a lattice from an n-best list of hypotheses. Since no costs are given by the recognizer, the language model development of Speech Dasher focuses on constructing word lattices and devising methods to assign probabilities to character sequences [7].

The language models developed for Speech Dasher are based on the n-best list obtained from an ASR engine. The n-best list contains a number hypotheses, but their rank is the only information available as to the probability the ASR engine assigns to those hypotheses. The lattice-based model, which is the one more relevant to the subject of this thesis, constructs a costless lattice by iteratively adding hypotheses to the lattice. The resulting lattice is expanded through *deletion* and other expansion methods before it is used as a language model. During its operation, *insertion* and *substitution* act as additional lattice expansion methods. The relative frequency of a given character at a given context and at the corresponding set of nodes in the lattice is used as a measure of probability.

The PPM model is used as a backup model. In case the lattice-based model fails at a certain point, PPM replaces it and it is used until the model can return to the lattice at suitable locations.

1.2. CATT

Dasher and Speech Dasher are the two applications that inspired the development of a computer-aided transcription tool (CATT). Dasher is taken as the graphical user interface of CATT. The language model of Dasher is replaced with a different model whose development and application is the main part of this thesis.

Text is entered letter by letter in Dasher. However, as seen in Figure 1.1, the most probable character sequences are displayed in such a way that groups of characters can be entered at once without navigating up and down to find the right letter. Such a smooth usage of Dasher requires a well trained language model. Since CATT incorporates the posterior probability distribution on alternate ASR hypotheses into the language model, it is expected that parts of the utterances that are covered in the lattice, which encodes the ASR hypotheses, will be easier to enter.

Speech Dasher makes use of an n-best list generated by an ASR engine. The idea of exploiting speech recognition hypotheses to create a language model is taken

from Speech Dasher [7]. Moreover, lattice expansion methods used in Speech Dasher are applied in CATT language models as well. However, Speech Dasher focuses on the personal use speech recognition via commercial recognizers. The speech recognizer used in Speech Dasher does not provide a probabilistic representation of the recognition hypotheses, therefore the ranking of hypotheses is the only source of information about the relative weights of the hypotheses assigned by the recognizer. CATT, on the other hand, utilizes the lattices generated by the ASR engine. These lattices are weighted finite-state acceptors that have a cost associated with each word in the lattice. Therefore, CATT directly uses the output of the acoustic and language models of the recognizer.

Unlike Speech Dasher, which applies lattice expansion methods during lattice construction before the model is used, CATT employs these methods on the fly when their usage help the models stay in the lattice or return to it.

Speech Dasher uses a backoff scheme to combine its lattice-based model and the PPM model. PPM is used in case the constructed lattice fails to provide the required probabilities. In this thesis, in addition to the backoff method, interpolation is also investigated as an alternative way of model building.

CATT combines two language models. A letter-based N-gram model is the default language model. The output lattices of an ASR engine are transformed into stochastic weighted finite-state acceptors and used as the main language model. In the following chapters of this thesis, the development of language models, the experiments conducted on letter-based N-gram language models, sentence level combination of models and CATT language models are explained, and the results are discussed.

2. SPEECH AND LANGUAGE PROCESSING

Speech is the main means of communication of people with each other, probably the only one for many people on earth. It is, therefore, a straightforward idea that humans may interact with machines through spoken language as well. Intense research going on since early 1970's in spoken language understanding area shows that making machines understand spoken language is a difficult task. However, significant progress has been done in this area. Setting attainable goals made development of many commercial automatic speech recognition (ASR) systems possible for specific domains and tasks. A greater variety of ASR systems are developed in the academia or other independent R&D institutions.

Speech recognition problem can be mathematically formulated as follows:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{A}) = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{W}|\mathbf{A}) \tag{2.1}$$

The goal is to find the most probable word sequence $\hat{\mathbf{W}}$ given the acoustic data $\mathbf{A}$. This formulation suggests that two probability values are necessary:

i. $P(\mathbf{A}|\mathbf{W})$, the probability that the speaker will generate the acoustic signal corresponding to acoustic data $\mathbf{A}$ given that he uttered the word sequence $\mathbf{W}$,
ii. and $P(\mathbf{W})$, the probability that the word sequence $\mathbf{W}$ will be uttered.

An ASR system has mainly two components: an acoustic processor utilizing an acoustic model and a linguistic decoder utilizing a language model. Acoustic modeling includes representing acoustic information, the acoustic waveform, in such a way that it may be used to discriminate different utterances. The Hidden Markov Model (HMM) is the most common acoustic model employed in speech recognizers [8]. A language model (LM) determines which word sequences are allowed to be spoken. It assigns probabilities to words and word sequences, so that it is possible to search for a certain word sequence within a dictionary and determine how probable it is that the speaker

uttered that word sequence.

Grammar and *parsing algorithm* are the two fundamental concepts of Chomsky's formal language theory [9], which used algebra and sets theory to define formal languages as sequences of symbols. With the availability of text corpora, it is possible to handle the problem from a statistical point of view by assigning accurate probabilities to word sequences which, otherwise, are assigned a probability value 1 or 0, if evaluated only by the rules of formal grammars [9].

Developing a language model includes various steps and requires many design decisions to be made and many trade-off issues to be resolved. Designing an ASR system for a specific domain allows putting constraints on the language model in terms of, for example, permitted grammatical structures or size of the vocabulary which includes the lexical entries defining the range of words the recognizer can generate as its output. On the other hand, a hypothetical all-purpose speaker-independent dictation system would require a very large vocabulary so that it could handle discourses on various topics.

Data-driven methods are widely used for language modeling. This requires text corpora with millions of words so that accurate probability distributions can be obtained for the basic recognition units. Words are taken as basic recognition unit for N-gram language models in most of the applications. N-gram models assume that the history of a given word includes only the $N-1$ previous words, hence the name N-gram. However, data sparseness is a major problem even for bigram models ($N = 2$). Many possible word combinations that may be used by a speaker may not be present or may not be well observed. The assigned probability extracted from the training data set for such a word combination would cause poor recognition performance. Since a word sequence may contain many N-grams, assigning zero probability to one of them would result in zero probability for the whole word sequence. Therefore many smoothing techniques are devised to manipulate the raw statistical model in such a way that high probabilities are lowered and very low probabilities are increased or assigned nonzero values to obtain a better performing system. Interpolation and backoff smoothing are

two main families of smoothing methods. Interpolated models incorporate lower order distributions into higher order distributions of all N-grams. However, backoff models use lower order information only for probability assignments to N-grams with zero counts. The way lower order probability distributions are used for setting higher order distributions is an important optimization problem in itself [8],[9], [10].

Due to the nature of the speech recognition task, some of the words in the vocabulary may be grouped according to some criterion to form a class. Incorporating these classes into N-grams may increase recognition performance since they model some words not present in the training set but present in test set if these words fall in one of the defined classes. Both rule-based and data-driven approaches are possible for *class N-grams* [9].

Vocabulary size is one of the most critical parameters. Fewer entries result in increasing out-of-vocabulary (OOV) rate because the coverage of domain specific vocabulary decreases, whereas more entries may cause more confusion since there are more possible combinations of words in this case, and the recognizer has to decide on the correct combination through a selection from among a greater number of candidates.

Selection of basic recognition units may be a critical design decision if using words in the model results in poor coverage of possible words to be recognized and high OOV rate. This may be the case for agglutinative or inflectional languages such as Turkish, Finnish or Czech [11].

The speech recognition problem can be represented via the finite-state automata framework. Various stages of an automatic speech recognizer can be represented as weighted or unweighted finite-state machines. Cascading these stages results in the recognizer [12].

Stochastic or statistical language models are based on the regularities of a language. These are learned from training data and encoded into the parameters of the

model, thus the model has assumptions on the form of the regularities of the language, and the parameters of the model correspond to mathematical descriptions of these regularities.

N-grams are widely used in language modeling [8], [9]. They are based on the idea that the probability of a given word depends on the words immediately preceding it. In general, not only the words but also sentences we utter depend on previous sentences we have uttered. What we say depends on what we have said up to that moment. Ideally, a speech recognizer may utilize the complete history of utterances of a person in recognizing the forthcoming utterances. However this a difficult and computationally expensive task. Truncating the history to a shorter context results in a practical and still useful method, N-gram language models.

N-gram language models assume that a given word depends only on the previous $N-1$ words as given by Equation 2.2.

$$P(w_i|w_1 w_2 \cdots w_{i-2} w_{i-1}) \cong P(w_i|w_{i-N+1}^{i-1}) \tag{2.2}$$

They are Markov models of order $N-1$ [9]. Base units other than words can be used in N-gram modeling. This is the case for agglutinative and highly inflectional languages. Selection of subword units as base units is an active research subject.

Data sparseness, or zero-frequency problem [6], is the central issue in N-gram language modeling. No matter how much data is collected, there will always be some N-gram contexts that aren't seen in training data. However a probabilistic model must give a nonzero probability to any word sequence because if an N-gram is given zero probability, the probability assigned to the whole word sequence becomes zero. Various smoothing methods have been developed to handle the zero-frequency problem.

Smoothing refers to transferring some of the probability mass from frequently observed contexts to rarely seen ones [10]. Backoff and deleted interpolation are the two main approaches for smoothing N-gram language models. Backoff models successively

shorten the context until they find a context that the model has seen before. Smoothing allocates some probability mass for this context shortening, i.e. backoff, event. On the other hand, deleted interpolation combines contexts of different orders.

Katz backoff is one of the most popular N-gram models. Some probability mass is allocated for unseen N-gram contexts so that it becomes possible to assign nonzero probabilities to such contexts by decreasing the order of the model. The probabilities of the lower order contexts are smoothed as well so that backoff to a lower order is again possible. This scheme prioritizes the longer contexts over the shorter ones [10]. In Equation 2.3, it is shown how the backoff scheme lowers model order when there are unseen contexts. $P^*(w_i|w_{i-N+1}^{i-1})$ is the smoothed probability for $P(w_i|w_{i-N+1}^{i-1})$. α is the backoff coefficient that adjusts the probability mass assigned to lower order N-grams. $CNT(w_{i-N+1}^{i})$ is the count of the N-gram context w_{i-N+1}^{i}.

$$P_{\text{Backoff}}(w_i|w_{i-N+1}^{i-1}) = \begin{cases} P^*(w_i|w_{i-N+1}^{i-1}) & \text{if } CNT(w_{i-N+1}^{i}) > 0 \\ \alpha(w_{i-N+1}^{i-1}) P_{\text{Backoff}}(w_i|w_{i-N+2}^{i-1}) & \text{otherwise.} \end{cases} \tag{2.3}$$

In deleted interpolation, N-grams of varying orders are linearly interpolated. The interpolation coefficients add up to 1. These coefficients are context dependent. To estimate these coefficients, first, the N-gram probabilities must be estimated using a portion of the training data. Then, the coefficients are estimated using the other portion, the held-out data [8]. In Equation 2.4, the deleted interpolation model is shown with context dependent coefficients λ_n.

$$\begin{aligned} P_{\text{DI}}(w_i|w_{i-N+1}^{i-1}) = {} & \lambda_1(w_{i-N+1}^{i-1}) P(w_i|w_{i-N+1}^{i-1}) \\ & + \lambda_2(w_{i-N+1}^{i-1}) P(w_i|w_{i-N+2}^{i-1}) \\ & \cdots \\ & + \lambda_N(w_{i-N+1}^{i-1}) P(w_i) \end{aligned} \tag{2.4}$$

The way the smoothing method is defined leads to various N-gram models. Good-

Turing discounting is based on the frequency of frequencies principle. The number contexts that have been seen c times is used to adjust the count of contexts that are seen $c-1$ times. Witten-Bell smoothing, on the other hand, uses the the count of the contexts that have been seen once to estimate the count of contexts never seen before. If T distinct symbols are seen N times so far, then the probability of seeing a novel symbol can be estimated by $\frac{T}{N}$. Another successful smoothing method is Kneser-Ney discounting. Although a word may appear very frequently in a corpus, it may be the case that this word appears only after a certain limited number of words. On the other hand, a less frequent word may appear in a greater number of contexts than the more frequent word. In this case, it may be expected that it is more likely that the less frequent word appears in a new context [9].

3. LANGUAGE MODELING

For the language model of CATT, individual letters are selected as base units. The Markov assumption, which is applied to sequences of words in Equation 2.2, is valid for letter-based N-grams as well, as shown in Equation 3.1. In addition to the letters in the Turkish alphabet, an additional symbol is included into the lexicon as a word-boundary marker.

$$P(c_i|c_1c_2\cdots c_{i-2}c_{i-1}) \cong P(c_i|c_{i-N+1}^{i-1}) \tag{3.1}$$

The language model used in CATT has two components:

i. the lattice generated by the ASR engine containing the recognition hypotheses,
ii. and a general letter N-gram language model that models the domain for which the ASR engine is developed.

The ASR has its own acoustic and language models and generates a lattice in form of a weighted finite-state acceptor. Each arc in the lattice has a word from the ASR engine's lexicon and a cost assigned to that word. The correct utterance may or may not be present in the lattice, moreover the hypothesis with the smallest cost, i.e. the highest probability, may not be the correct utterance. Nevertheless, the lattice contains the most relevant information about the correct utterance. Making use of this information as much as possible for efficient transcription of the correct utterance is one of the main objectives of language modeling for CATT.

The lattice encodes a limited number of utterances. The one that is uttered by the speaker may not be among those utterances in the lattice. In this case, the transcriber should be able to enter a word or part of a word not present in the lattice. The purpose of the letter N-gram language model is to provide the transcriber with appropriate alternatives given the present context.

3.1. Lattice Manipulation

Word-based lattice generated by the ASR engine must be transformed into a letter-based lattice with word boundary symbol inserted between words. AT&T's FSM Library provides the necessary tools for lattice manipulation [14].

Using a 50K word lexicon, a finite-state transducer was created to expand each word in the lexicon. If a WFSA with one or more words from the lexicon on paths leading to final nodes is composed with the word-to-letter FST and the resulting WFST is projected onto the output side, the output WFSA has a series of nodes connected with arcs labeled with a single symbol (a letter or the word-boundary symbol) corresponding to the word in the input WFSA that is expanded. The letters of the Turkish alphabet, word-boundary symbol and some other symbols necessary for other lattice manipulation operations form the lexicon for the output WFSA. This new lexicon is also used for letter N-gram language models.

The original lattice contains words on its arcs. Words with common prefixes and/or roots are on separate arcs with, in general, different costs associated with them and the lattice is deterministic. However, once this word-based lattice is transformed to a letter-based lattice, it is possible that the lattice is no more deterministic. Determinization and minimization operations generate a deterministic and compact lattice [15].

Another issue to be resolved is related to the fact that the lattice will be used as a language model. Hence it must have a proper probability distribution over all of the possible paths that are present in the lattice. This is not the case because the lattice reflects a small part of the ASR engine's search space over which all possible word sequences add up to one. The costs in the transformed lattice are pushed and the residual costs are removed so that the resulting lattice becomes stochastic.

The manipulation steps mentioned above will be illustrated with a simple example.

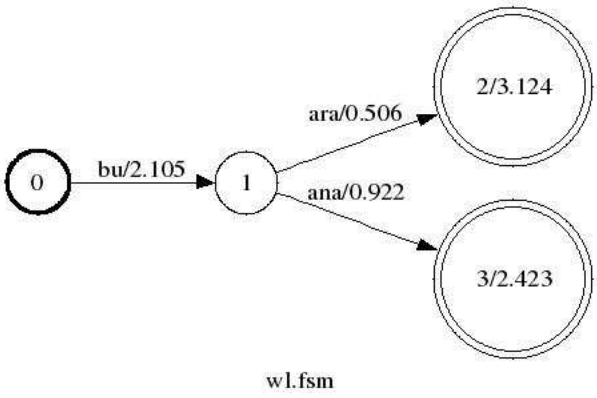

Figure 3.1. Word-based Lattice

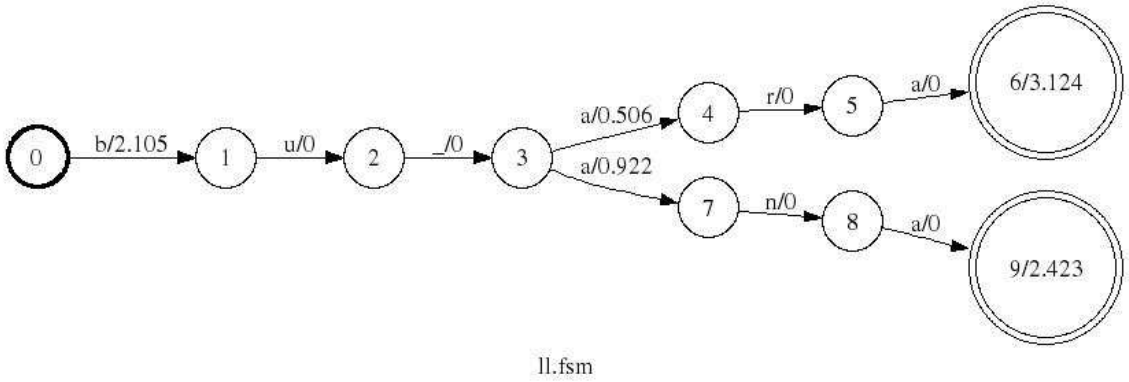

Figure 3.2. Letter-based Lattice

The word-based lattice in Figure 3.1 is deterministic but not stochastic. The costs are the negative of the natural logarithm of the probability values. For example, the cost assigned to the string `bu ara` is 5.735, which corresponds to the probability 0.00323. The cost of the other possible string, "`bu ana`", that is encoded in the lattice is 5.45 ($\equiv 0.0043$). The sum of these probabilities is not 1 as it should be for a stochastic lattice. Without any calculation, noting the nonzero cost of the word "`bu`" is sufficient to show that it is not stochastic.

In Figure 3.2, the lattice after the transformation into a letter-based lattice is shown. The cost structure is not changed. Each of the possible paths gets the same total cost assigned as before. Note that the lattice is no more deterministic due to the arcs going out of the node 3 with the same letter.

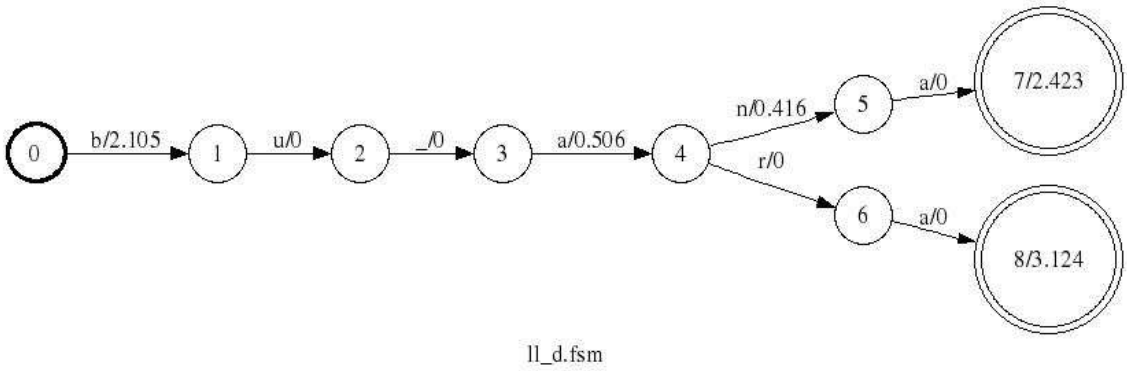

Figure 3.3. The lattice after determinization

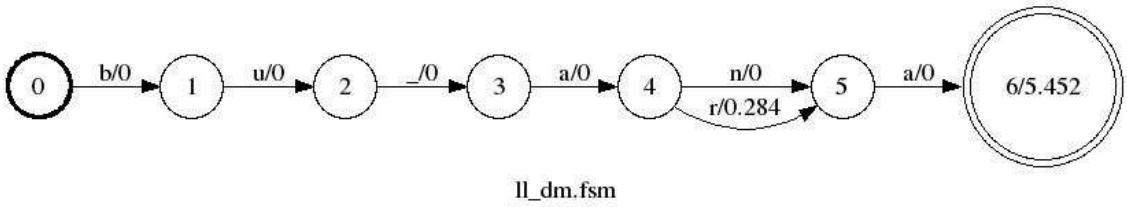

Figure 3.4. The lattice after minimization

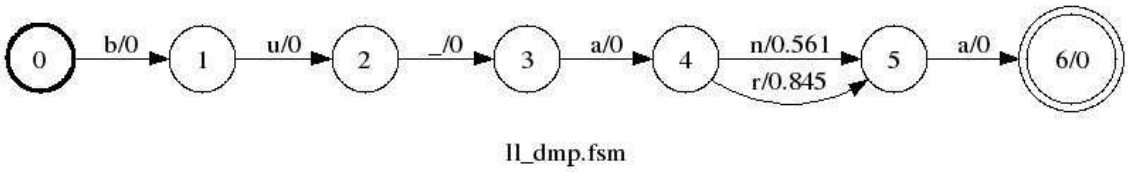

Figure 3.5. The lattice after pushing

After determinization, the arcs with the same letter in Figure 3.2 are combined as shown in Figure 3.3. However, the cost structure is still preserved, i.e. the lattice is still not stochastic.

Minimization step unifies the arcs going into the final nodes since they carry the same symbol (Figure 3.4). Although the lattice looks quite different than the original lattice, the costs of the paths are the same, however the costs of the arcs are manipulated and the residual cost is placed at the final node.

Push operation makes the lattice stochastic (Figure 3.5). The sum of the probabilities at node 4 is $e^{-0.561} + e^{-0.845} = 1$. The costs of the other characters is 0 since there are no alternatives to those characters. Note that a cost of 0 corresponds to the probability value of 1 ($e^{-0} = 1$).

3.2. Lattice Expansion

The output lattice of the ASR engine contains the recognition hypotheses. In order the recognition to be correct, the most likely hypothesis must be the utterance to be recognized. However, the utterance may be encoded in the lattice but may not be the most likely hypothesis. Moreover, it is possible that the utterance is not within the hypotheses. Since CATT's language model is based on the ASR output, it must be able to handle such cases where the utterance is missing in the lattice. To tackle this problem, we take the approach of expanding the lattice in case it fails to align with the transcription entered by the transcriber and we aim to present the most likely letters given the current context based on the lattice.

The basic idea behind lattice expansion is the assumption that differences between the utterance and the hypotheses in the lattice can be accounted for by *deletion*, *insertion* and *substitution*. These editing operations are defined with respect to the hypotheses in the lattice because CATT's usage as transcription tool translates to editing the output of an ASR engine with help of a general language model.

If the current word in the utterance is not aligned with any of the possible words, the *active* words, in the lattice but is aligned with at least one of the words following the active words, the current word in the utterance can be aligned with the lattice by *deleting* the active words in the lattice. *Deletion* is considered only when the alignment fails at the beginning of a new word.

If the word following the current word in the utterance aligns with the lattice, then alignment can continue if the current word in the utterance is omitted. This operation is called *insertion* because deleting the current word in the utterance is equivalent to *inserting* that word into the lattice.

Another source of failure is the case when the current word in the utterance is not aligned with any of the active words in the lattice, but the next word in the utterance aligns with at least one of the words following the active words. Thus alignment can continue by *substituting* the active words with the current word in the utterance. *Substitution* can be viewed as a combination of deletion and insertion operations.

Insertion and substitution operations require that we know the word following the current word in the utterance. This is not possible at the time of failure. Thus, whether the alignment can continue through one of these editing operations is determined once the next word in the utterance is reached. However, since we know which words follow the active words in the lattice, deletion can be decided on at the instant of failure.

Lattice expansion methods will be illustrated with a simple example on Figure 3.6. To focus only on the operations themselves, the costs on the arcs are removed. The N-gram language model is not shown but assumed to exist and provide probabilities.

Assume that the utterance to be aligned with the lattice is "`a e`". `e` doesn't align with either `b` on arc from node 1 to 2, i.e. (1→2), or with `c` on arc (1→3). If we *delete* arc (1→2), the current letter can be aligned with the arc (2→5).

Assume that the utterance to be aligned with the lattice is "`a h c f`". `h` doesn't

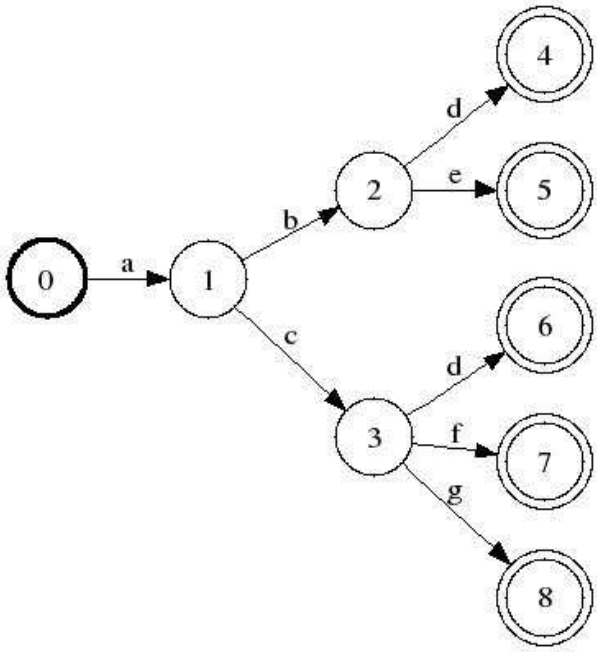

Figure 3.6. Example lattice to illustrate lattice expansion methods

align either with `b` or with `c`. If we *insert* an arc (1→1) carrying `h`, the rest of the string can be aligned with (1→3) and (3→7).

Assume that the utterance to be aligned with the lattice is "`a h g`". `h` doesn't align either with `b` or with `c`. If we *substitute* the letter the arc (1→3) carries with `h`, the rest of the string can be aligned with (3→8).

3.3. Letter N-Gram Model

The lattice, which is used as a language model based on the utterance, is complemented with a letter-based N-gram language model. Depending on how it is combined with the lattice, i.e. which combination scheme is chosen, it can improve the language model in one of the two ways:

i. In the backoff model, when the lattice fails to provide a probability for a given character, the N-gram model is used as the probability model.
ii. In the interpolation model, N-gram model is always used regardless of whether

the lattice provides probabilities or not. N-gram model smooths the probabilities given by the lattice by making the combined model allocate some probability mass for the characters not covered by the lattice.

The probabilistic model the lattice provides is not smoothed in any way because it only contains the "guesses" of the ASR engine. Due to the same reason, the lattice does not contain any information about any utterance other than the one to which it corresponds. But a language model should cover as many words and word combinations as possible.

It is necessary to train the N-gram model from the same domain as the utterances to be transcribed. Thus the model can provide more relevant alternatives and better probability estimates. To prevent that the model becomes outdated with time, it may be updated regularly or adapted to the recently entered data using CATT.

The letter N-gram language model used in CATT is trained using the AT&T's GRM library [13]. This library provides tools for training and manipulating N-gram language models. Backoff and interpolation are the two models supported by the library. Besides the default Katz discounting method, absolute or Kneser-Ney discounting may be used.

The order, model, discounting method, training data size are some parameters that has to be decided on in building the N-gram model. Experiments on these parameters are described in the following chapter.

3.4. Combining the Lattice and the N-Gram Model

Combining the lattice with the N-gram model must address the following issues:

- The lattice contains more relevant information about the utterance to be transcribed. Therefore, the probabilities provided by the lattice should be pronounced.

- Since recognition errors are inevitable, the stochastic lattice must be modified to be able to assign nonzero probabilities to character sequences not covered by the lattice.
- Both models must be combined in such a way that the resulting model is still a stochastic one.

Word-based lattices are transformed to letter-based lattices, and the resulting lattices are converted to deterministic and stochastic WFSAs that can be used as language model. In addition to lattices, a letter N-gram model is built, which is also a stochastic model. *Backoff* and *interpolation* are the two approaches taken to combine the lattice and the letter N-gram model.

In the backoff scheme, the lattice is taken to be the main model. In case it fails to provide probabilities for a given context, the model backs off to the letter N-gram model. The model tries to return to the lattice as soon as possible. If the letter N-gram model were taken as the main model, there would not be a need to back off to the other model because the N-gram model is based on letters instead of words, thus *Out of Vocabulary* (OOV) rate is 0, and, as long as no unknown character is given, it provides a probability for any n-gram. Moreover, since the lattice contains the hypotheses of an ASR engine, the probabilities given for a context are higher than those given by the letter N-gram model.

In the interpolation scheme, the two models are linearly interpolated to obtain a smooth probability distribution. Thus the letter N-gram model is used even if the lattice can provide probabilities. Once the lattice fails to do so, the N-gram model remains as the only model. The model tries to enter the lattice as soon as possible as in the backoff scheme.

Both models employ the aforementioned lattice expansion methods. It is these methods that determine when and if the lattice will be left and reentered.

3.4.1. Overview

In the subsequent parts of this section, backoff and interpolation models will be explained in detail. These explanations describe the complete model together with lattice expansion methods.

Here, an overview of both models will be given. In order to make clear how lattice expansion operations modify the models, the forms of the models before and after adding these operations are shown.

The backoff model without lattice expansion can be summarized as follows:

$$P(c_i|h_i) = \begin{cases} \lambda \cdot P_L(c_i|n_i) & \text{in lattice} \\ (1-\lambda) \cdot P_N(c_i|c_{i-N+1}^{i-1}) & \text{lattice fails} \\ P_N(c_i|c_{i-N+1}^{i-1}) & \text{in N-gram} \end{cases} \tag{3.2}$$

c_i is the current character to be aligned with the lattice and h_i is the history of the characters and the corresponding nodes. n_i is a node in the lattice or in the N-gram model. λ is the probability mass allocated to the successful alignment of c_i with the lattice, hence $P_L(c_i|n_i)$ is the probability given to c_i by the lattice, where L stands for ***L***attice. Note that P_L is conditioned on n_i since the current node completely describes the position of the model in the lattice. P_N is the probability assigned to c_i by the ***N***-gram model. P_N is conditioned on c_{i-N+1}^{i-1} due to the Markov assumption.

As long as the characters in the utterance are aligned with the lattice, the first equation is valid. If the lattice fails at a character, the model leaves the lattice by taking the $1-\lambda$ path and backs off to the N-gram model. Subsequent characters are covered by the N-gram model until the utterance is finished. Since the lattice is not an alternative after backoff, there is no need to allocate any probability mass for re-entrance event at any point in the N-gram model, hence the probabilities obtained from the N-gram model are not weighted by a coefficient. Once the model backs off, there is no way it can return back to lattice. The information it contains is not used

after a failure event.

The interpolation model without lattice expansion can be summarized as follows:

$$P(c_i|h_i) = \begin{cases} \lambda \cdot P_L(c_i|n_i) + (1-\lambda) \cdot P_N(c_i|c_{i-N+1}^{i-1}) & \text{in lattice} \\ (1-\lambda) \cdot P_N(c_i|c_{i-N+1}^{i-1}) & \text{lattice fails} \\ P_N(c_i|c_{i-N+1}^{i-1}) & \text{in N-gram} \end{cases} \tag{3.3}$$

In this model, λ is the interpolation coefficient. No matter if alignment of the utterance with the lattice successfully continues or not, the N-gram model is always used in the model. As long as the lattice provides probabilities, both models are interpolated. If the lattice fails at a point, the model starts to use only the N-gram model. Note that, at the point of failure, there are, if a final node in the lattice is not reached, some characters other than c_i which the lattice could have covered if the current character was one of those characters. Therefore, the N-gram is still weighted although it is the only model that provides a probability. However, for the remaining characters, the model is completely out of lattice and the probabilities obtained from the N-gram model are not weighted.

Since the lattice is more relevant to the utterance to be transcribed than the N-gram model, not exploiting the lattice after a failure event, which may very well occur at the beginning of an utterance, is an undesirable situation. Staying in the lattice despite failures, or returning to lattice at a later point, if leaving it in the case of a failure is unavoidable, would be a solution to this problem.

Lattice expansion methods may help the combined model to make use of the lattice as much as possible. Deletion may prevent that the model leaves the lattice in case of a failure at the beginning of a word by aligning the current character with the initial letter of the words following the active word in the lattice. On the other hand, if the model leaves the lattice, insertion or substitution or both can make the model reenter the lattice at a word boundary.

Since lattice expansion methods are applicable at certain points during the alignment process, only the modified parts of the Equations 3.2 and 3.3 are given below.

In case of DELetion, the probability distribution at failure points in the lattice are calculated as follows:

Backoff

$$P(c_i|h_i) = (1-\lambda) \begin{cases} c_{DEL} \cdot P_L^{DEL}(c_i|n_i) & DEL \\ (1-c_{DEL}) \cdot \alpha \cdot P_N(c_i|c_{i-N+1}^{i-1}) & \text{backoff} \end{cases} \tag{3.4}$$

Since the lattice has failed, $1-\lambda$ path is taken. c_{DEL} is the probability mass allocated for the deletion event. $P_L^{DEL}(c_i|n_i)$ is the probability calculated from the lattice at the beginning of the next words in the lattice. If deletion is not possible, the model backs off to the N-gram model. $1-c_{DEL}$ is the probability mass allocated from $1-\lambda$ for the backoff event. P_N, the probability obtained from the N-gram model, is weighted by a coefficient α because the characters seen at the point of failure, at node n_i, and at the beginning of next words must be excluded from possible characters the N-gram can cover so that the sum of the probabilities for the characters that may be covered by the N-gram in this specific failure event add up to 1.

Interpolation

$$P(c_i|h_i) = (1-\lambda) \cdot P_N(c_i|c_{i-N+1}^{i-1}) + \begin{cases} c_{DEL} \cdot P_L^{DEL}(c_i|n_i) & DEL \\ 0 & \text{can't do } DEL \end{cases} \tag{3.5}$$

In case of interpolation, there is no need for α since the N-gram model is always used regardless of whether deletion operation is accomplished or not. If deletion is not possible, $(1-\lambda)P_N(c_i|c_{i-N+1}^{i-1})$ is the probability value.

In case of INSertion or SUBstitution, the probability distribution at possible re-

entrance points, at word boundaries reached in the N-gram model, back to the lattice are calculated as follows:

Backoff

$$P(c_i|h_i) = \begin{cases} \gamma \cdot P_L^*(c_i|n_i) & INS \text{ or } SUB \\ (1-\gamma) \cdot \alpha \cdot P_N(c_i|c_{i-N+1}^{i-1}) & \text{can't return} \end{cases}$$

γ is the probability mass allocated for the re-entrance event. How P_L^* is computed depends on whether insertion or substitution is possible, . If re-entrance is not possible, the probability from the N-gram model is not only weighted by $1-\gamma$, it is also adjusted by α by excluding the characters seen in the lattice from the N-gram.

Interpolation

$$P(c_i|h_i) = (1-\gamma) \cdot P_N(c_i|c_{i-N+1}^{i-1}) + \begin{cases} \gamma \cdot P_L^*(c_i|n_i) & INS \text{ or } SUB \\ 0 & \text{can't return} \end{cases}$$

The only difference from the backoff model is that the probability obtained from the N-gram is not affected whether lattice expansion can be performed or not.

For the sake of completeness, backoff and interpolation models are explained below in greater detail without omitting what has been already mentioned in the overview. Therefore there may be some repetitions.

3.4.2. Backoff Model

The backoff model uses the lattice as the main model. The coverage of the lattice is limited, and, occasionally, it will fail to align with the utterance. However the sum of the probabilities that are assigned to all possible paths in the lattice add up to 1. This means that no probability mass is reserved for cases when the lattice fails and has to be left via backing off to the N-gram model. Such a failure can occur at any point

in the lattice. Thus, the probability distribution at each point in the lattice must have some probability mass kept aside for failure situations.

A similar situation occurs after backing off to the N-gram model. The backoff model aims to reenter the lattice as soon as possible. Until such a possible entrance point is reached, the N-gram model provides the necessary probability values. However, similar to the issue with the lattice, the N-gram model doesn't reserve any probability mass for the event of leaving it. Such an event is not included in the lexicon but imposed by the backoff scheme. Unlike the lattice, the leaving points in the N-gram model, or the re-entrance point back to the lattice, are only at word boundaries, thus a context dependent manipulation of the probability distribution is necessary for proper probability mass assignment to the re-entrance event.

The backoff model requires another probability distribution adjustment. At the points of failure in the lattice, the model backs off to the N-gram model. The characters starting the paths at the current position in the lattice are also present in the N-gram model. However these characters have to be excluded from probability calculations. The probability of the current character in the utterance given by the N-gram model must be normalized by the sum of the probabilities of the characters in the N-gram model different than the active characters in the lattice. A similar calculation must be performed at re-entrance points in the N-gram model whenever there are some candidates in the lattice for re-entrance but none of them match the current character in the utterance. In this case, the backoff model continues to use the N-gram model, however all those candidate characters in the lattice must be excluded from the calculation of the probability value obtained from the N-gram model.

After this introduction to the backoff model, a detailed explanation for probability calculations at different points in the model will be given below. This model may also act as a starting point for the development of other backoff models. Depending on a specific implementation of the backoff model, the mathematical expressions below may be evaluated differently. Some implementations will be described in the following chapter.

$P(c_i|h_i)$, the probability of a character c_i given the history h_i, can be calculated for different contexts as follows:

1. If $c_{i-1}, n_i, c_i \in$ Lattice

$$\lambda(h_i)\, P_L(c_i|n_i) \tag{3.6}$$

The previous character c_{i-1} is successfully aligned with the lattice and the model is at the node n_i of the lattice. Moreover, the current character c_i is aligned with the lattice as well. There is no need for lattice expansion or backing off to the N-gram model. Since the lattice is deterministic and stochastic, the cost of the character c_i is the probability $P_L(c_i|n_i)$. The probability $P(c_i|h_i)$ is computed only from the lattice. Since the node n_i completely describes the current position of the model in the combined search space of the lattice and the N-gram model, $P(c_i|h_i) = P(c_i|n_i)$. However $\sum_{c_k \in C(n_i)} P(c_k|n_i)$, i.e. the sum of the probabilities $P(c_k|n_i)$ where c_k is a character on an arc leaving the node n_i, equals to 1. $C(n_i)$ is the set of all characters on arcs leaving the node n_i. $P_L(c_i|c_i \notin C(n_i))$, the probability assigned by the lattice to a character c_i not on any of the arcs leaving the node n_i, is therefore 0. However, the model must assign a nonzero probability to any character in the lexicon for any context that may occur. Hence P_L is lowered by λ so that a probability mass $1 - \lambda$ for the failure event $c_i \notin C(n_i)$ is reserved. λ is a function of the history, thus its value can be assigned in a context dependent manner or, as it will be seen in the next chapter, it can be a constant.

2. If $c_{i-1}, n_i \in$ Lattice $\wedge$ $c_i \notin C(n_i)$ $\wedge$ $c_{i-1} \neq \#$

$$(1 - \lambda(h_i))\, \alpha(\Phi(n_i))\, P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.7}$$

α is a normalization factor as explained before. $\Phi(n_i)$ is the set nodes that are directly accessible from n_i. Since n_i is not a node at the beginning of a word, $\Phi(n_i) = \{n_i\}$. To simplify notation, $\{n_i\}$ will be shown as only n_i except for set union operations. This is one of the two alternatives to case 1. $c_i \notin C(n_i)$, i.e. c_i

is on none of the arcs leaving the node n_i. This is a failure event. However, the failure occurs not after a word boundary or at the beginning of a new word but within a word or at the end of a word, hence the condition $c_{i-1} \neq \#$. This means that the lattice expansion method deletion is not an option since it is considered only at word boundaries. The model backs off to the N-gram model or leaves the lattice and enters the N-gram model. The probability mass allocated to this event is $1 - \lambda$. As before, λ is a function of the history. The probability $P(c_i|h_i)$ is computed from the letter N-gram model. Since the position of the model in the N-gram model depends on the context c_{i-N+1}^{i-1}, $P(c_i|h_i) = P_N(c_i|c_{i-N+1}^{i-1})$. However, this probability value cannot be used as it is. The combined model backs off to the N-gram because the current character c_i is not covered by the lattice given the history h_i. Unless n_i is a final node, there is at least one character c_k which is covered by the lattice. If c_i were c_k, then no failure would occur. Note that the N-gram model can assign a probability $P_N(c_k|c_{i-N+1}^{i-1})$ for any character in the lexicon, i.e. also to the characters $c_k \in C(n_i)$. Also note that $\sum_{c_k \in \text{Lexicon}} P_N(c_k|c_{i-N+1}^{i-1}) = 1$. However the N-gram model is needed only for the characters $c_k \notin C(n_i)$. Hence probability $P_N(c_i|c_{i-N+1}^{i-1})$ must be normalized by the sum of the probability values the N-gram model assigns to all characters not covered by the lattice at the given context. This normalization is performed by α. The calculation of α is given in Equation 3.15. In this case, $\alpha(A(n_i)) = \alpha(n_i)$. Note that if $C(n_i) = \emptyset$, the denominator becomes 1, hence $\alpha = 1$. α is a function of a set of nodes in the lattice. If $A(n_i)$ is an empty set, α is directly set to 1.

3. If $c_{i-1}, n_i \in \text{Lattice} \ \wedge \ c_i \notin C(n_i) \ \wedge \ c_{i-1} = \#$

$$(1 - \lambda(h_i))\, P^{\dagger}(c_i|h_i) \tag{3.8}$$

This is the second alternative to case 1. The conditions for this case are identical to those of case 2 except the failure occurs at a word boundary. The lattice expansion method deletion may be performed so that the model remains in the lattice without backing off to the N-gram model. Remaining in the lattice is preferred over backoff because the probability values the lattice gives are, in general, greater

than those given by the N-gram. The computation of the probability $P^{\dagger}(c_i|h_i)$ is rather involved. Before going into the computational details, the possible courses of action at such a failure point in the lattice will be given:

i. $N(n_i) = \emptyset$: The set of nodes at the beginning of the next words in the lattice, i.e. $N(n_i)$, is empty. There are no words following the active word in the lattice. Deletion is not an option. The model backs off to the N-gram.

ii. $N(n_i) \neq \emptyset \ \wedge \ c_i \in C(N(n_i))$: There are words following the active word in the lattice and c_i matches at least one of the characters $C(N(n_i))$. In this case, deletion is possible. However, the calculation of $P^{\dagger}(c_i|h_i)$ must be carried out over the set of characters $\{c_k : c_k \in (C(N(n_i)) \setminus C(n_i))\}$, i.e. over the characters that are on arcs leaving the nodes $N(n_i)$ but are different than the characters $C(n_i)$. The reason is the same as the reason for exclusion of some characters in the calculation of α in case 2.

iii. $N(n_i) \neq \emptyset \ \wedge \ c_i \notin C(N(n_i))$: There are words following the active word in the lattice. However, none of the characters c_k on arcs leaving the nodes at the beginning of these next words $N(n_i)$ matches c_i. In this case, again, the combined model backs off. The calculation of the normalization factor α is more complicated than in case 2. Not only the characters $C(n_i)$ but also $C(N(n_i))$ must be excluded in the calculation of α, i.e. the set of characters $C(W(n_i)) = C(\Phi(n_i) \bigcup N(n_i)) = C(\{n_i\} \bigcup N(n_i))$.

Since the possibilities and the notation are explained above, the calculation of $P^{\dagger}(c_i|h_i)$ can be given without any further explanations:

$$P^{\dagger}(c_i|h_i) = \begin{cases} \alpha(\Phi(n_i))\, P_N(c_i|c_{i-N+1}^{i-1}) & \text{if } N(n_i) = \emptyset \\ c_{DEL}\, P_L^{DEL}(c_i|N(n_i)) & \text{if } N(n_i) \neq \emptyset \wedge c_i \in C(N(n_i)) \\ (1 - c_{DEL})\, \alpha(W(n_i))\, P_N(c_i|c_{i-N+1}^{i-1}) & \text{if } N(n_i) \neq \emptyset \wedge c_i \notin C(N(n_i)) \end{cases} \tag{3.9}$$

where

$$P_L^{DEL}(c_i|N(n_i)) = \begin{cases} \frac{\sum_{n_k \in N(n_i)} P_L(c_i|n_k)}{\sum_{c_k \in (C(N(n_i)) \setminus C(n_i))} \sum_{n_k \in N(n_i)} P_L(c_k|n_k)} & \text{if } c_i \in (C(N(n_i)) \setminus C(n_i)) \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

c_{DEL} is the probability mass allocated to the deletion event.

4. If $c_{i-1}, n_i \notin$ Lattice $\wedge$ $c_{i-1} \neq \#$

$$P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.11}$$

The combined model has already backed off to the N-gram model. The current node n_i is in the N-gram model. The previous character is not the word boundary symbol, hence insertion or substitution is not possible. $P(c_i|h_i)$ is equal to the probability the N-gram model gives to c_i without any normalization or probability mass allocation.

5. If $c_{i-1}, n_i \notin$ Lattice $\wedge$ $c_i \in C(\Phi(n_i))$ $\wedge$ $c_{i-1} = \#$

$$\gamma(h_i)\, P_L^*(c_i|\Phi(n_i)) \tag{3.12}$$

γ is the probability mass for the re-entrance event. $P_L^*(c_i|\Phi(n_i))$ is the probability distribution that is defined on the fly, depending on the nodes and characters "seen" in the lattice. In this case, the model returns back to the lattice. After the failure in the lattice, depending on the position of the failure, a word or part the rest of a word is covered by the N-gram. The previous character is the word boundary symbol. The current character is "seen" in the lattice from the node n_i, i.e. it is at least on one of the arcs leaving the nodes that are accessible from n_i through either insertion or substitution operation. Note that n_i is in the N-gram model, not in the lattice. There are nodes in the lattice which are accessible from n_i through insertion or substitution without consuming any characters. These nodes in the lattice, if they exist, form $\Phi(n_i)$. In order to explain how the re-

entrance back to the lattice occurs, it should be made clear how the set of nodes $\Phi(n_i)$ "seen" from the node n_i is formed. If the failure event that resulted in backing off to N-gram model at a previous point in the lattice has occurred at a word boundary, i.e. if $l(h_i) = b(h_i)$, such a node is a possible re-entrance candidate and is "seen" from n_i. $l(h_i)$ is the node n_k in the lattice at which a failure event occurred and the model backed off to the N-gram model. $b(h_i)$ is a function that gives the node n_k in the lattice and in the history h_i closest to the current node n_i such that the character $s(n_k) = c_{k-1} = \#$. $b(h_i)$ gives only nodes from the lattice, so it doesn't return the current node n_i which is also at a word boundary and $s(n_i) = \#$. If $N(l(h_i)) \neq \emptyset$, then these nodes are accessible from n_i as well. γ is defined as a history dependent parameter. The computation of P_L^* depends on how $\Phi(n_i)$ is defined. Note that this case assumes $c_i \in C(\Phi(n_i))$.

$$P_L^*(c_i|\Phi(n_i)) = \begin{cases} P_L(c_i|b(h_i)) & \text{if } c_i \in C(b(h_i)) \wedge N(b(h_i)) = \emptyset \\ c_{INS}\, P_L(c_i|b(h_i)) & \text{if } c_i \notin C(N(b(h_i))) \wedge N(b(h_i)) \neq \emptyset \\ c_{SUB}\, P_L(c_i|N(b(h_i)) & \text{if } c_i \notin C(b(h_i)) \wedge N(b(h_i)) \neq \emptyset \end{cases} \tag{3.13}$$

where $c_{SUB} = 1 - c_{INS}$. All the conditions above must be considered with the initially given condition that the combined model returns to the lattice. Hence, for example in the second condition of the above equation, if c_i is not among the characters that are candidates for a substitution, then it must be among those that are candidates for an insertion. Other conditions are interpreted similarly.

6. If $c_{i-1}, n_i \notin$ Lattice $\wedge$ $c_i \notin C(\Phi(n_i))$ $\wedge$ $c_{i-1} = \#$

$$(1 - \gamma(h_i))\, \alpha(\Phi(n_i))\, P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.14}$$

This case is the alternative to case 5. A re-entrance is not possible. α is computed as in case 2.

$\alpha(A(n_i))$ is defined as follows:

$$\alpha(A(n_i)) = \begin{cases} \frac{1}{\sum_{c_k \notin C(A(n_i))} P_N(c_k|c_{i-N+1}^{i-1})} = \frac{1}{1-\sum_{c_k \in C(A(n_i))} P_N(c_k|c_{i-N+1}^{i-1})} & \text{if } A(n_i) \neq \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{3.15}$$

In the denominator of the first fraction, the summation is carried out over all characters not in the set of characters in the lattice that are seen at the node n_i. These are all of the characters in the lexicon other than $C(A(n_i))$. Since $\sum_{c_k \in \text{Lexicon}} P(c_k|c_{i-N+1}^{i-1}) = 1$, the same probability mass can be computed in a easier way as given in the denominator of the second fraction.

3.4.3. Interpolation Model

The interpolation model does not prefer one model over the other one. The models are combined via linear interpolation. When the lattice fails, the N-gram model provides the probability values until the interpolation model can enter the lattice again.

How $P(c_i|h_i)$, the probability of a character c_i given the history h_i, can be calculated for different contexts is given below. However, since detailed explanations are given above for the backoff model, only the differences between the interpolation model and the backoff model will be made clear.

1. If $c_{i-1}, n_i, c_i \in$ Lattice

$$\lambda(h_i)\,(1 - c(n_i))\,P_L(c_i|n_i) \;+\; (1 - \lambda(h_i))\,P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.16}$$

λ acts as an interpolation coefficient. It defines how the probability mass is divided between the lattice and the N-gram model. In order the model to be as generic as possible, λ is defined as a function the history. $c(n_i)$ is defined as

follows:

$$c(n_i) = \begin{cases} c_{DEL} & \text{if } c_{i-1} = \# \wedge N(n_i) \neq \emptyset \wedge (C(N(n_i)) \setminus C(n_i)) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

$c(n_i)$ allocates some probability mass for the deletion operation. Three conditions must be met so that it is assigned a nonzero value:

i. The previous character must be the word boundary symbol, i.e. the model must be at a word boundary.

ii. There must be at least one word following the active word in the lattice.

iii. At least one of the first characters of the next words must be different than the characters at the current position in the lattice.

These conditions must be evaluated in the order they are given.

2. If $c_{i-1}, n_i \in \text{Lattice} \ \wedge \ c_i \notin C(\Phi(n_i)) \ \wedge \ c_{i-1} \neq \#$

$$(1 - \lambda(h_i))\, P_N(c_i | c_{i-N+1}^{i-1}) \tag{3.18}$$

Lattice fails at a position other than word boundary. Since deletion is not possible and backing off is not an option for the interpolation model, only the N-gram model provides some probability. However, it is still weighted by $1 - \lambda$ because $C(n_i)$ for the node n_i in the lattice is not empty. If the current character was one of the characters in $C(n_i)$, then the probability would be computed according to case 1. Hence a probability mass of size λ must be kept aside so that characters in $C(n_i)$ can be covered when needed.

3. If $c_{i-1}, n_i \in \text{Lattice} \ \wedge \ c_i \notin C(\Phi(n_i)) \ \wedge \ c_{i-1} = \#$

$$\lambda(h_i)\, P_L^{\dagger}(c_i | n_i) \ + \ (1 - \lambda(h_i))\, P_N(c_i | h_i) \tag{3.19}$$

The computation of $P_L^{\dagger}$ is similar to Equation 3.9 in case 3 of the backoff model:

$$P_L^{\dagger}(c_i|h_i) = \begin{cases} 0 & \text{if } N(n_i) = \emptyset \\ c_{DEL}\, P_L^{DEL}(c_i|N(n_i)) & \text{if } N(n_i) \neq \emptyset \wedge c_i \in C(N(n_i)) \\ 0 & \text{if } N(n_i) \neq \emptyset \wedge c_i \notin C(N(n_i)) \end{cases} \tag{3.20}$$

$P_L^{DEL}(c_i|N(n_i))$ is computed as in the Equation 3.10.

4. If $c_{i-1}, n_i \notin$ Lattice $\wedge$ $c_{i-1} \neq \#$

$$P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.21}$$

This equation is identical to Equation 3.11 although the models are different. In this case, the combined model is out of the lattice. There are no characters that can be covered by the lattice, hence there is no need to allocate any probability mass for the lattice.

5. If $c_{i-1}, n_i \notin$ Lattice $\wedge$ $c_{i-1} = \#$

$$\gamma(h_i)\, P_L^*(c_i|\Phi(n_i)) \; + \; (1 - \gamma(h_i))\, P_N(c_i|c_{i-N+1}^{i-1}) \tag{3.22}$$

This is like a combined version of the cases 5 and 6 in the backoff model. Since there is no need for α, the events of successful re-entrance and failure to do so can be given as a single case. $P_L^*(c_i|\Phi(n_i))$ is calculated similar to Equation 3.13.

$$P_L^*(c_i|\Phi(n_i)) = \begin{cases} 0 & \text{if } c_i \notin C(\Phi(n_i)) \\ P_L(c_i|b(h_i)) & \text{if } c_i \in C(b(h_i)) \wedge N(b(h_i)) = \emptyset \\ c_{INS}\, P_L(c_i|b(h_i)) & \text{if } c_i \notin C(N(b(h_i))) \wedge N(b(h_i)) \neq \emptyset \\ c_{SUB}\, P_L(c_i|N(b(h_i))) & \text{if } c_i \notin C(b(h_i)) \wedge N(b(h_i)) \neq \emptyset \end{cases} \tag{3.23}$$

As before, $c_{SUB} = 1 - c_{INS}$.

3.4.4. Practical Aspects of Lattice Expansion

The transcription of an utterance is basically an alignment operation of the utterance with the lattice and the N-gram model. In practice, the worst that can happen is that the model leaves the lattice and does not return thereafter. If the N-gram model is trained on sufficiently large data and it has a proper model order, it can function as a suitable substitute for the combined model until the rest of the utterance is transcribed. However, the lattice provides a better estimate of the utterance to be transcribed. To make use of the lattice as much as possible, lattice expansion methods are used to locally warp the lattice and enable continued alignment, as in the case of deletion, or re-entrance back to lattice, as in the cases of insertion and substitution. However, if a deletion operation fails and, after covering the rest of the word with the N-gram model, an insertion operation is performed, then the part of the word which was already aligned up to the failure point would be used again.

Consider how $P^{\dagger}(c_i|h_i)$ in the Equation 3.8 is defined in Equation 3.9. In case of a deletion operation, the probability P_L^{DEL} is computed with respect to characters $c_k \in (C(N(n_i)) \setminus C(n_i))$, i.e. the characters which occupy the arcs going out of the nodes at the beginning of the next words in the lattice but are different than the characters already visible at the node of failure n_i. The conditions in Equation 3.9 and 3.10 clearly show that deletion operations is performed if c_i matches any of the characters in $C(N(n_i))$. Such a criterion for performing deletion is rather weak. The average word in the text corpus used in the training of N-gram models has 6.4 letters. A successful match of the first character can not guarantee that the rest of the characters will match as well. The same reasoning applies to insertion and substitution.

A remedy to this problem would be looking ahead in the utterance and checking whether a deletion, insertion or substitution operation will be accomplished. Lookahead in the lattice is not an issue because we have full knowledge of the lattice before starting the transcription. In such a case, the probability calculations in backoff and interpolation models above must be changed to include the lookahead operation. Especially the calculation of the backoff normalization coefficient α becomes more compli-

cated. Not only the characters beginning next words but all of the next words starting with characters other than the ones seen at the failure node would be excluded from the N-gram. This is also valid for insertion and substitution. Models applying this lookahead method are experimented in the next section.

Note that looking ahead in the utterance is not possible in practice. The characters entered by the transcriber must be used to determine which path in the lattice will be taken or whether the lattice should be left. Therefore, the combined model used in CATT must be based on the original model definition which considers only the initial letters of words, instead of complete words, to decide on whether a lattice expansion operation should be attempted.

4. EXPERIMENTS

CATT combines two models: a lattice-based language model utilizing the posterior probability distribution on alternate ASR hypotheses and a letter-based N-gram language model. These models are combined at letter level instead of word level because CATT uses the Dasher application as its graphical user interface which is operated by continuous mouse gestures and lets users to enter text letter by letter. Several models to combine the lattice and the letter N-gram model are devised. To compare model performances, model parameters are experimentally optimized. To find out whether the models generalize to unseen data, parameters optimized on a test set are used on another test set.

Although researchers in the speech and language processing field widely use word-based N-gram language models, the subject of this thesis requires letter-based modeling. Since such an approach dramatically shrinks the lexicon size, issues that arise in case of word-based modeling are expected to be of less importance for letter N-gram modeling. Three sets of experiments on letter N-gram modeling aim to provide insights on how models devised for word-based models perform as letter-based models and what parameters affect the performance of these models. Due to limited size of the lexicon, contexts of greater orders may be practical. Moreover, data sparseness problem is not as important as in word-based modeling due to relatively small number of contexts that can be generated by the lexicon. Hence, the effect of decreasing training data size on model performance is an interesting subject to investigate.

The Speech Dasher application uses n-best lists of recognition hypotheses to construct lattices on which a letter-based language model is defined. This approach is similar to the way CATT uses the recognition results for language modeling. The model that gives the best results in Speech Dasher is implemented to find out whether utilizing the posterior probability distribution provided by the speech recognizer brings any gain.

Table 4.1. Statistics of the text corpus

Number of Sentences	931,511
Number of Words	11,596,492
Number of Characters	85,075,303
Number of Letters	74,410,322
Words per Sentence	12.45
Letters per Sentence	79.88
Letters per Word	6.42

Experiments performed by combining the lattice and letter N-gram models at sentence level and by using the N-gram models without combining with the lattice provide baseline results with which the performance of the combined models can be compared.

4.1. Statistics of the Test Data

Experiments are performed on Turkish read news material. 2 sets of ASR outputs are used as test data in these experiments. The first set contains 553 utterances read by a female speaker, the second 480 utterances recorded from the "news for the hearing impaired" broadcast news. The ASR system that is being developed in the BUSIM Laboratory of Boğaziçi University with software provided by AT&T Labs - Research generates WFSAs containing a number hypotheses for the utterance. The number of hypotheses may differ greatly from utterance to utterance.

A number of experiments have been conducted on letter N-gram models to investigate the effects of model type, model order and training data size on model performance. A 5-gram Kneser-Ney smoothed model is chosen for CATT combined language model experiments. This model is trained on 650,642 words with 4,174,047 letters.

The statistics of the text corpus used in the N-gram model experiments are summarized in Table 4.1. The corpus contains texts collected from Turkish newspapers.

Table 4.2. Statistics of the test data for CATT language model experiments

	Data Set 1	Data Set 2
Number of Utterances	553	480
Number of Words	6989	2248
Number of Characters	51973	16991
Number of Letters	45537	15223
Words per Utterance	12.6	4.68
Letters per Utterance	82.3	31.71
Letters per Word	6.5	6.77
Lattice Utterance Error Rate	29.3%	38.1%
1-Best Utterance Error Rate	75.9%	59.8%
OOV Rate	10.2%	8.8%
Lattice WER	12.8%	10.3%
1-Best WER	32.2%	26.2%

This corpus is partitioned into various portions to be used as training and test data in letter N-gram model experiments.

Some statistics about the test sets is given in Table 4.2.

4.2. Letter N-Gram Model Experiments

N-grams, as a powerful statistical language processing tool, are devised to be used with words or with linguistic units of comparable length. Various smoothing techniques proposed in the literature aim to address the data sparseness problem. Human languages contain tens of thousands of words. Through suffixation and compounding, the number of wordforms increase significantly compared to lemmas. This is especially true for agglutinative languages [10]. However the language model used in CATT is based on individual characters instead of letters. Since the ASR outputs normalized text in lower case, the lexicon of the N-gram language model to be used consists of the

letters of the alphabet, a word-boundary symbol and a few special symbols used only for proper calculation of N-gram probabilities but not during CATT's operation. The number of possible character combinations are much lower than that of word combinations even for moderate lexicon sizes. For example, lattices obtained from the ASR engine uses a lexicon with 50K entries. This corresponds to more than $125 \cdot 10^{12}$ trigrams. On the other hand, 29 letters of the Turkish alphabet and a word-boundary symbol can generate at most 27000 trigrams or $243 \cdot 10^5$ (24.3 million) 5-grams. 100K sentences from the text corpus used in training the N-gram models contain about 1.3 million words but more than 9.5 million characters.

Since it is practically impossible to collect enough text that covers all possible wordforms for an agglutinative language, it seems to be possible to build a successful letter-based N-gram models for CATT's purposes with very small text corpora compared to those used for word-based models.

The following experiments are performed to investigate how the model type, model order and training data size affect the performance of letter-based N-gram models. Cross entropy is the metric used to quantify the model performances, although improvement in CE does not imply improvement in WER. It is generally assumed that lower entropy correlates with better performance of models in applications [8], [10], [16].

Cross entropy is calculated as follows:

$$\mathrm{CE} = \frac{1}{\sum_{k=1}^{T} N_k} \sum_{k=1}^{T} \sum_{i=1}^{N_k} -\log_2 P(c_i|h_i) \tag{4.1}$$

where T is the number sentences in the test data and N_k the number of characters in the k-th sentence.

Table 4.3. Statistics of training and test data for model selection experiments

	Training Data	Test Data
Number of Sentences	92849	104833
Number of Words	1155122	1303252
Number of Characters	8464858	9567203
Number of Letters	7402585	8368784
Words per Sentence	12.4	12.4
Letters per Sentence	79.7	79.8
Letters per Word	6.4	6.4

Table 4.4. Results of N-gram model selection experiments

	BO Katz	BO Kneser-Ney	DI	DI Kneser-Ney
CE (bits/character)	2.091	2.087	2.108	2.092

4.2.1. N-Gram Model Selection Experiments

AT&T's GRM Library [13] is used to train the letter N-gram models. Experiments with Katz backoff, Kneser-Ney smoothed backoff, deleted interpolation (DI) and Kneser-Ney smoothed deleted interpolation models are carried out [8], [9], [10] .

The statistics of the training and test data are summarized in Table 4.3. The corpus contains texts from Turkish newspapers. Test and training data are uniformly selected from the corpus, thus their statistics are very similar. 5-gram models are used in this experiment.

The results are summarized in Table 4.4. The cross entropy values show slight differences. However, backoff models performed better than deleted interpolation models. The following experiments are performed with backoff models.

Table 4.5. Statistics of the training and test data for model order experiments

	Training Data	Test Data
Number of Sentences	838662	92849
Number of Words	10441370	1155122
Number of Characters	76610445	8464858
Number of Letters	67007737	7402585
Words per Sentence	12.4	12.4
Letters per Sentence	79.9	79.7
Letters per Word	6.4	6.4

Table 4.6. Results of model order experiments

Model order (n-gram)	BO Katz (CE in bits/character)	BO Kneser-Ney (CE in bits/character)
6	1.853	1.875
5	2.053	2.047
4	2.422	2.416
3	3.000	2.992
2	3.614	3.605

4.2.2. N-Gram Model Order Experiments

Backoff N-gram models of varying order (from bigram to 6-gram) are compared in this experiment. Table 4.5 shows the statistics of the training and test data for this experiment. As in the previous, both sets are formed by uniform selection from the corpus.

The results of this experiment are given in Table 4.6. Although differences due to discounting methods are very small, model order affects the performance significantly. Increasing the model order by one decreases the cross entropy 15% on the average.

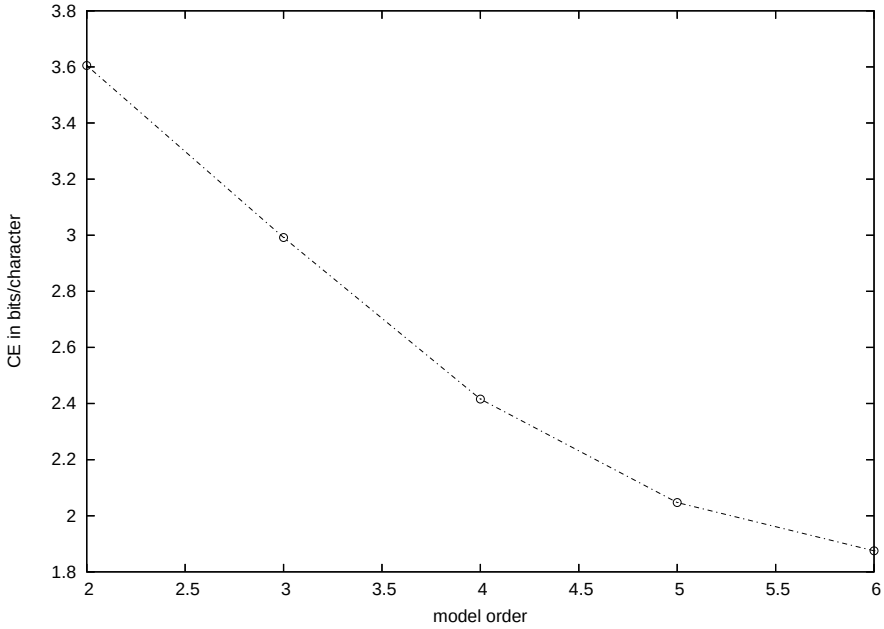

Figure 4.1. CE versus model order for KN smoothed backoff models

Figure 4.1 depicts the variation in CE with respect to increasing model order. The relative gain obtained by increasing the model order decreases after the order 4.

Considering the practical usage of CATT, resource requirements of the application should be taken into account as well. Figure 4.2 shows the change in the logarithm of the number of states in FSM representation of Kneser-Ney smoothed backoff models with increasing model order.

Although the 6-gram model outperforms the others as expected, it requires a ca. 57MB model file in binary format, whereas the 5-gram model is stored in a 15MB binary file.

4.2.3. Training Data Size Experiments

It was intended to design an adaptive model in addition to the basic N-gram model so that the transcribed utterances could be used to improve the CATT's language model. Such a model would alleviate the problem of outdated N-gram model since the topics in the news change rapidly from day to day. In order to make use

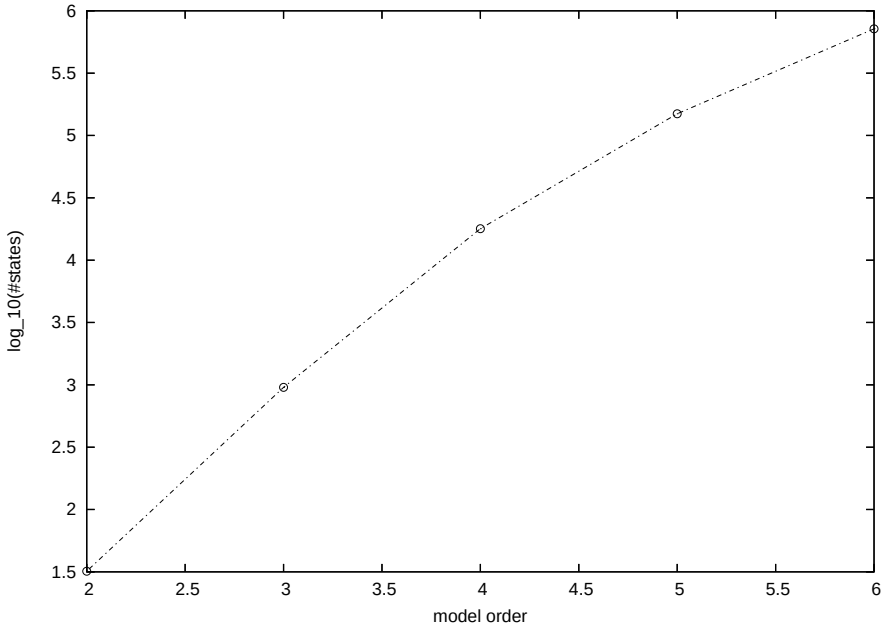

Figure 4.2. Number states versus model order for KN smoothed BO models

of the transcriptions, there has to be a measure to quantify the importance of the new in-domain data with respect to the training data which is very large in size but not as relevant to the contents of the current speech data as the recently transcribed utterances.

Another motivation for these experiments lies in the fact that, as opposed to word-based N-gram modeling, data sparseness is not a critical issue in letter-based N-gram modeling since the number of possible contexts for a given context is limited. Hence the effect of decreasing training data size on model performance is an interesting issue in itself.

The most straightforward measure is the data size, in number of characters in case of CATT. The purpose of the training data size experiments is to determine an effective size of the training data that can be taken as a parameter to weight a basic N-gram model and an adaptive model based on recent transcriptions. Although an adaptive model has not been implemented, the results of these experiments can be used as a starting point for future research and development efforts.

Table 4.7. Statistics of the training data for training data size experiments

Training Data Set	Number of Characters	Relative Size wrt Data Set 10
1	297303	0.004
2	589435	0.008
3	1183591	0.015
4	2370379	0.031
5	4772272	0.062
6	9567203	0.125
7	19150358	0.250
8	38340884	0.500
9	57476293	0.750
10	76610445	1.000

In training data size experiments, data sets of varying sizes have been used to train 4- and 5-gram, Katz and Kneser-Ney backoff models. Test data is the same as in the N-gram model order experiments. Table 4.7 shows the data sizes of the 10 data sets prepared together with the normalized size of each set with respect to the first data set. These values may be of use in evaluating the performances of the models with varying training data sizes.

As seen in Table 4.8, training data size has little effect on the cross entropy per character. Although the first model's training data is about 250 times the 10th model's training data, it improves the cross entropy over the the 10th model by only 14%.

Figure 4.3 shows the change in CE with decreasing training data size. However, a more insightful analysis of the effect of training data size can be made by inspecting the Figure 4.4. In this figure, the cross entropy of the test data is plotted against the number of the sentences in log scale.

Table 4.8. Results of training data size experiments for KN-BO model

Set \ Model Order	4-gram (CE in bits/char.)	5-gram (CE in bits/char.)
1	2.543	2.362
2	2.497	2.266
3	2.467	2.191
4	2.443	2.137
5	2.431	2.101
6	2.424	2.078
7	2.420	2.062
8	2.417	2.053
9	2.416	2.050
10	2.416	2.047

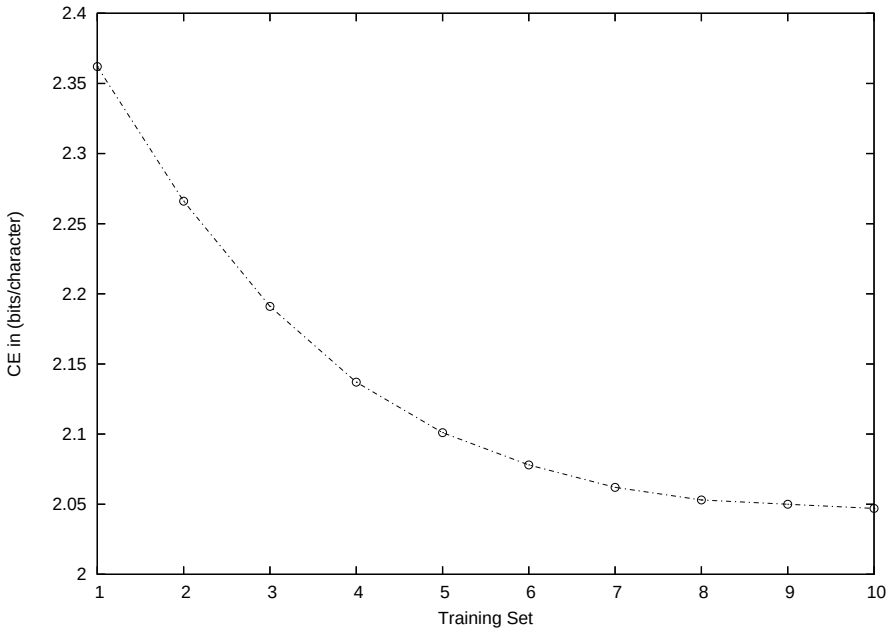

Figure 4.3. CE versus training data sets for 5-gram KN-BO model

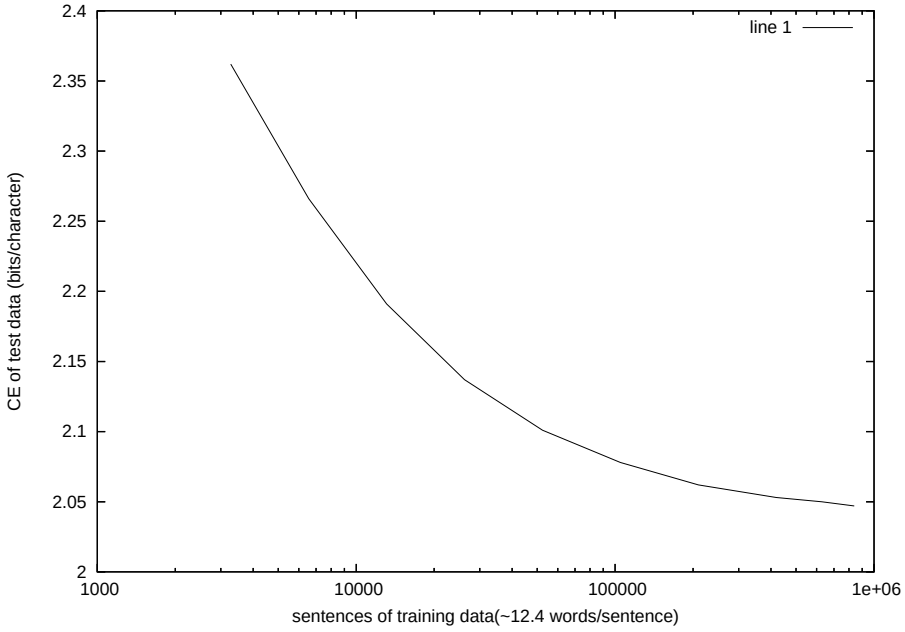

Figure 4.4. CE versus training data size for 5-gram KN-BO model

4.3. Sentence Level Experiments

The lattice and the N-gram language model are combined at letter level (LLC). In order to see whether it performs better than sentence level composition (SLC), a number of experiments are carried out.

CE for both of the test data sets are calculated from the transformed lattices combined with a letter 5-gram language model and a letter 6-gram language model.

The lattice and the N-gram models are combined by backoff and interpolation methods. In the backoff scheme, if the lattice contains the utterance, it assigns a probability to the utterance. This value is weighted by a parameter λ as in the LLC case to allocate a probability mass for the failure event. If the utterance is not encoded in the lattice, than the probability assigned by the N-gram model is taken with weight $1 - \lambda$. In the interpolation scheme, λ is the interpolation coefficient. If lattice fails to provide a probability value, the one provided by the N-gram model is still weighted by $1 - \lambda$ because the lattice contains at least one hypothesis and a probability mass must be allocated for those hypotheses.

Table 4.9. Results of SLC experiments with Test Set 1

	BO (CE in bits/char.)	IP (CE in bits/char.)
5-gram	0.885	0.885
6-gram	0.779	0.779

Table 4.10. Results of SLC experiments with Test Set 2

	BO (CE in bits/char.)	IP (CE in bits/char.)
5-gram	1.076	1.076
6-gram	0.951	0.951

As seen in Tables 4.9 and 4.10, the results for both methods are identical. However, increasing the model order of the N-gram model decreases the cross entropy, i.e. increases the performance of the combined model. Since the results seem independent of the combination method, plots for the change in CE with lambda for the test set 1 is given only for the interpolation model. Figure 4.5 shows the results of the experiment with the letter 5-gram language model, and Figure 4.6 with the letter 6-gram model.

4.4. CATT Language Model Experiments

A general framework to combine the lattice and the letter N-gram model was defined in the previous chapter. That framework, depending on which combination scheme is preferred, lists the states at which the combining model can be during transcription. The parameters that are specified in the model are not explicitly defined but given as functions of the history. Moreover, the probability distributions are history-dependent as well. The combining model may be taken as a template and converted into a practical model by defining how the given parameters are set, how the probability distributions handle lattice expansion, and what other parameters, if any, are introduced within these distributions.

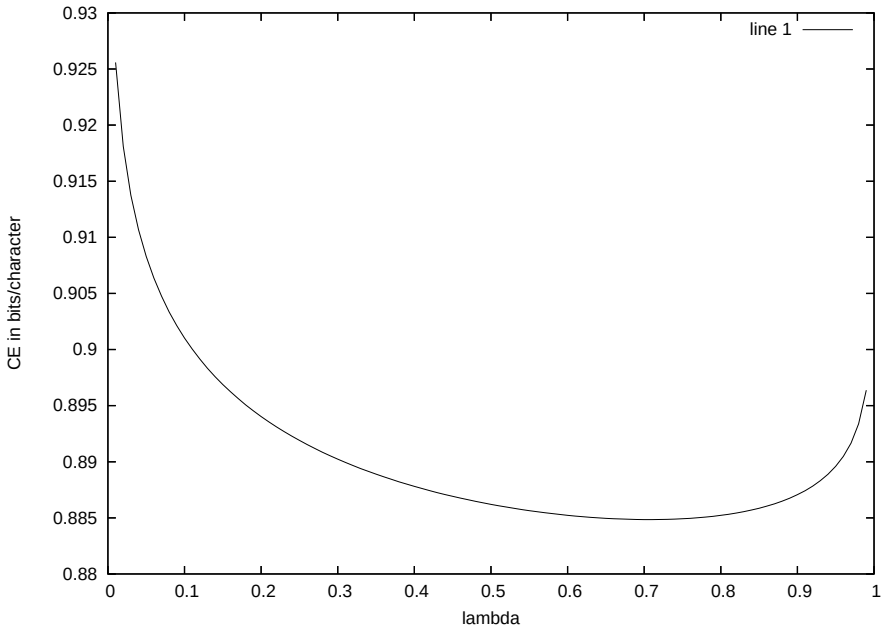

Figure 4.5. CE versus λ for SLC via IP with 5-gram model

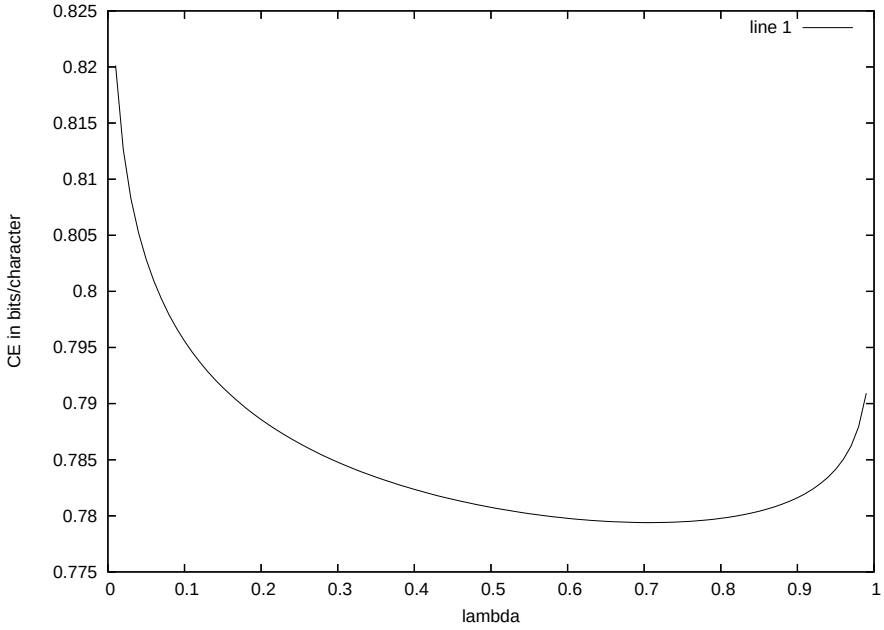

Figure 4.6. CE versus λ for SLC via IP with 6-gram model

In this section, we define some of the models that are designed based on the template given in the previous chapter, and we give the results of the experiments that are performed with these models. Some details on how these models are programmatically implemented for testing and what modifications are made for their use with Dasher are given in the following chapter.

4.4.1. Backoff

The backoff scheme described in the previous chapter with equations from 3.6 through 3.14 is used as a basis to design different language models. These will be explained here, and results of the experiments to find the best model parameters will be given.

4.4.1.1. Backoff-Multiple Presence (BOMP). This model uses three parameters: λ, γ and c_{DEL}. Insertion and substitution editing operations are combined into a single reentrance event to avoid introducing an additional parameter in addition to γ for that event. Moreover, all of the lattice expansion operations may be carried out over more than one path if possible since we cannot know which path is going to be successfully complete the editing operation.

The explanations below about the details of the model correspond to the cases 1 through 6 of the backoff model in Chapter 3 respectively:

1. A constant λ value is used. It is the probability mass allocated for remaining in the lattice. P_L is computed based on n_i, the node which the model has reached so far. However, during one of the lattice expansion operations, the model can be at more than one nodes at the same time. Then, P_L is the sum of the costs of arcs, leaving one of the active nodes, with the character c_i over sum of the costs of all arcs leaving all of the active nodes. So we have a proper probability distribution.
2. Lattice fails at a character other than the beginning of a word. In this case we

leave the lattice with $1 - \lambda$, i.e. we back off to N-gram model, and weight P_N with α such that P_N is divided by the sum of costs of the characters other than those seen in the lattice at the node of failure. Hence we allocate a probability mass of 1 to all possible paths after leaving the lattice as it should be.

3. If lattice fails at the beginning of a word, there are two scenarios to consider: Either we stay in the lattice via deletion or we back off to N-gram model. However, if there are no words weight following the current word, then the only option is backing off to N-gram model. In this case, calculation of α exclude only the characters seen at n_i in the lattice. The BOMP model assumes that lattice expansion operations should be performed if it is certain that the operation will be finished successfully. Hence, the model tries to align the current word in the utterance with all of the words following the active word weight. If at least one of the next words align with the current word, then deletion operation is performed. In this case $P^{\dagger}$ is the multiplication of c_{DEL} and $P_L(c_i|N(n_i))$. P_L is defined in the Equation 3.9. If none of the words align completely, we back off to N-gram model, but calculation of α is more complicated. We take a list of all characters seen at n_i, the node of failure. Excluding these, we determine all the words following the active word in the lattice, thus we make a second list. α is calculated by excluding the costs given by the N-gram model to the characters in the first list and to words in the second list. The reason for making the second list with words following the active word in the lattice instead of the characters at the beginning of those next words is the following: If the second list consisted of the characters at the beginning of the next words, then we would leave out all possible words aligning partially with any of the next words. This corresponds to a greater probability mass excluded from calculation of α than the criterion for not performing deletion suggests. In such a case, we wouldn't have a proper probability distribution. The way α is calculated here differs from the Equation 3.15.
4. P_N is calculated from the N-gram model since we are out of the lattice and not at a word boundary.
5. If a word boundary is reached while being out of the lattice and if the current word aligns with at least one of the candidates in the lattice, then the model reenters

Table 4.11. Experimental result for BOMP model

	Test Set 1
λ	0.996
γ	0.626
c_{DEL}	0.110
CE in bits/char.	0.266274

the lattice at all possible entry points, not only at those that align completely. γ is the probability mass allocated for this re-entrance event. P_L^* is computed similar to Equation 3.13, but insertion and substitution are combined in this model, so there are no separate cases for these lattice expansion methods. $\Phi(n_i) = W(l(h_i)) = \Phi(l(h_i)) \bigcup N(l(h_i))$, i.e. the set of nodes consisting of the node of failure before backing off to N-gram ($l(h_i)$) and the nodes at the beginning of words following the active at the point of failure before backing off to N-gram ($N(l(h_i))$). If this the second word covered out of lattice, then $\Phi(n_i)$ may also contain $N(N(l(h_i)))$.

6. This is the alternative to case 5. If re-entrance is not possible, the model continues to cover the current word in N-gram. α is calculated along the lines of the previous cases.

4.4.1.2. Backoff-Multiple Presence 2 (BOMP2). This model is the same as the BOMP model above except that the parameter c_{DEL} is removed from the model. Although deletion operation is ruled out as a possible lattice expansion method, it is investigated whether a simplification of the model outweighs the cost incurred by backing off to a more costly model, i.e. the N-gram model. This may be the case if deletion operation is rarely performed.

4.4.1.3. Backoff-Geometric (BOGEO). Geometric distribution assumes a binary success-failure event. A series of success events are terminated by a failure event or vice versa. This approach is similar to the failure event we define for the lattice in case the current

Table 4.12. Experimental result for BOMP2 model

	Test Set 1	Test Set 2
λ	0.996	0.987
γ	0.622	0.526
CE in bits/char.	0.26788	0.54855

Table 4.13. Experimental result for BOGEO model

p	0.864
q	0.970
γ	0.626
c_{DEL}	0.110
CE in bits/char.	0.813537

character in the utterance doesn't align with the lattice. However, most of the times, the lattice aligns with the words in the utterance without failure. In order to apply the geometric distribution approach to the combined model, the parameter λ is defined as two separate parameters p and q. p is the probability of success, i.e. the alignment of the current character with the lattice within the word. $1-p$ is the probability of failure event which is either the event of word termination (successful word alignment) or a real failure (misalignment of the utterance with the lattice). q is the probability of the word termination event, $1-q$ that of the real failure event. Besides this redefinition of λ, this model is identical to the BOMP model.

4.4.1.4. Backoff-CATT. The models described above are designed for testing purposes. This affects how the lattice expansion methods are realized. In these models, the lattice is expanded if the expansion will be successful. Since the tests are run offline, the models can look ahead, for example, before attempting deletion. If there is a complete alignment, deletion is performed to remain in the lattice.

Table 4.14. Experimental result for BOCATT model

	Test Set 1	Test Set 2
λ	0.994	0.982
γ	0.675	0.772
c_{DEL}	0.323	0.200
CE in bits/char.	0.30482	0.58723

However, such a mechanism is not possible during real usage because it is not possible to look ahead. To simulate this situation, the criterion for performing lattice expansion is relaxed to the alignment of the first letter of a word. If, in case of failure at a word boundary, the first letter of one of the next words in the lattice matches to the current letter, the combined model performs a deletion operations and jumps to all of the matching nodes. Since a successful deletion operation requires the alignment of all of the letters, a significant amount of the attempted deletion operations are bound to fail at some point within the word. The same situation applies to insertion and substitution. Due to increased number of failures, more than necessary jumps between the models incur additional costs. Therefore, this model may perform worse than some of the models experimented with.

4.4.2. Interpolation

The interpolation scheme described in the previous chapter with equations from 3.16 through 3.22 is used as a basis to design different language models. These will be explained here, and results of the experiments to find the best model parameters will be given.

4.4.2.1. Interpolation-Multiple Presence (IPMP). This model is similar to BOMP model with respect to its parameters and the application of lattice expansion method. However, the N-gram model always provides probabilities to the combined model. In this scheme, $1 - \lambda$ is not the probability mass reserved for the failure event but the inter-

Table 4.15. Experimental result for IPMP2 model

	Test Set 1	Test Set 2
λ	0.990	0.972
γ	0.495	0.344
CE in bits/char.	0.26237	0.53906

polation coefficient for the N-gram model.

To avoid repetitions, the points for which the explanations for the BOMP model are not sufficient will be made clear.

2. Since this model doesn't back off to the N-gram model, α calculation is not necessary.
3. If deletion is possible $P_L^\dagger = c_{DEL} \cdot P_L(c_i|N(n_i))$, else it is 0.
5. $\gamma = 0$ if $\Phi(n_i) = \emptyset$, or it is a constant determined experimentally.

4.4.2.2. Interpolation-Multiple Presence 2 (IPMP2). In the IPMP experiments, the c_{DEL} parameter was forced to be 0. This model is the revised version of IPMP model. c_{DEL} is removed as in the case of BOMP2 model. The only difference with respect to the IPMP model is in case 3, where, now, $P_L^\dagger$ is always 0.

4.4.2.3. Interpolation-CATT. This is a modified version of IPMP2 along the sames lines as Backoff-CATT. Lattice expansion operations are attempted based only on the alignment of the first letter of a word.

An additional experiment which aims to find out how the CATT language models perform on a new test set after the parameters are optimized on a test set. The results can be compared with optimum values obtained on the new test set to determine how the models generalize to unseen data.

Table 4.16. Experimental result for IPCATT model

	Test Set 1	Test Set 2
λ	0.990	0.963
γ	0.468	0.528
CE in bits/char.	0.29419	0.56694

4.5. Speech Dasher Language Model Experiments

CATT language model is inspired from Speech Dasher (SD). ASR output is exploited in the SD language model. Moreover, lattice expansion methods are employed to recover word errors in the lattice. In order to compare the performances of CATT and SD language models, the methods developed must be applied to the same test data.

Since SD works on n-best lists obtained from a commercial speech recognizer, constructing lattices from n-best lists on which a language model is defined is one of the main parts of the Vertanen's work [7]. The lattice-based language model of SD is based on lattices which are constructed from the n-best lists via an iterative algorithm that tries to keep the resulting lattice as compact as possible. Lattice expansion methods are applied on this compact lattice to recover from possible recognition errors. The lattice-based model is very similar to the backoff scheme applied in CATT. Although the constructed lattice is word-based, it is used in a letter-based fashion. Therefore, it is not deterministic. Since there are no costs in the lattice, the frequency of different symbols that are seen at the locations the model has reached during the alignment process are used to define a probability distribution. The alignment process is followed by partial paths that reach from the root of the lattice to the locations the model has reached within the lattice. Failure of the lattice results in paths dying at the point of failure. Each dying path gives birth to children pointing to the beginning of the word and to the words following the current word. Once the current word reaches the end-of-word symbol, these new paths are used as possible entry points back to

lattice. Although each path has an initial probability of 1, paths created through lattice expansion methods are given costs less than 1 as a penalty so that the original lattice which encodes the hypotheses licensed by the n-best list is given more weight. After recovering from a word error, the paths are rebuilt from the start node instead of from the position where all paths died. This is expected to prevent that the model is driven to a part of the lattice which poorly aligns with the rest of the utterance. During errors, probability values are provided by the PPM model. However, although the initial part of a word may have failed to align with the lattice, the ending part may do so. To make use of such cases, a so called fringe-based estimation is devised which is activated only after a preset number of symbols in the utterance align with the ending of the words in the lattice which failed recently. Another method that weights the trust put in the PPM model, digression detection method, is employed whenever consecutive word errors occur.

Lattice expansion and probability estimation strategies of SD are different than the strategies of CATT. In SD, the lattice constructed from the n-best list is initially expanded before it is used as a language model. Since the lattice is constructed as a word-based lattice but used as a letter-based model, it may not be deterministic at some points. Therefore, the probability estimation includes all the paths that point to the current character to be entered by the user, and normalizes the costs of these paths by the sum of the costs of all active paths. It may be the case that the character entered by the user is covered by the original lattice before expansion and that none of the partial paths pointing to the parts of the final lattice added after expansion covers that character. Then the original lattice would be de-emphasized, since the probability estimation takes all partial paths into account, including the paths pointing to those nodes added due to expansion, to normalize the sum of costs of successfully aligning paths. On the other hand, CATT expands the lattice, if the lattice does not cover the character to be entered. If it aligns with that character, then the probability given to that character is calculated only from the original unexpanded lattice.

However, SD and CATT handle the re-entrance to the lattice after word errors in a similar way. Language models of both applications consider the beginning of the

word during which the alignment failed and the words following this word as possible locations for the re-entrance event.

According to experimental results, lattice-based language model outperforms a string-based language model which is not mentioned here. Although fringe-based estimation improves cross entropy to some extent, the effect of digression detection was minimal. Among many lattice expansion methods devised, one word insertion expansion, which corresponds to the deletion operation in terms of the terminology of CATT, is found to be the one that reduces the cross entropy significantly [7]. The language model implemented for comparison experiments includes deletion, insertion and substitution methods and fringe-based estimation. The parameter values given by Vertanen are used in the experiments. Lattice expansion methods other than one word insertion expansion are left out due to their less significant contribution to model performance. Digression detection is not implemented due to the same reason. The PPM model is replaced with N-gram models differing in order and training data size. A 6-gram model trained on 73.9 MB text and 5-gram models trained on 4.6 MB and 294 KB text data are used in the experiments. The PPM model used in SD language model is trained on 300 KB of text.

4.6. Experimental Results

In the experiments, Kneser-Ney smoothed 5-gram model trained with data set 5 in Table 4.7 is used. This model assigns 2.140 bits/character to the test data set 1 and 2.194 to the test data set 2 for the CATT language model experiments. The 6-gram model built for training model order experiments assigns 1.859 bits/char and 1.936 bits/char. for test data sets 1 and 2 respectively. Parameters in the combined models were optimized through exhaustive search. CE per character is used as the cost metric to be optimized.

Tables 4.11 through 4.16 show the optimal parameter values along with the optimal cost value. Table 4.17 compares the performances of the experimented models. Each of the models outperform the 5-gram model, and even the 6-gram model by a large

Table 4.17. Results for CATT language model experiments

Model	Test Set 1 CE in bits/char.	Test Set 2 CE in bits/char.
BOMP	0.266	-
BOMP2	0.268	0.549
BOGEO	0.814	-
BOCATT	0.305	0.587
IPMP2	0.262	0.539
IPCATT	0.294	0.567
SD (6-gram 73.9MB)	0.452	0.599
SD (5-gram 4.6MB)	0.485	0.639
SD (5-gram 294KB)	0.516	0.662
SLC (6-Gram)	0.779	0.951
SLC (5-Gram)	0.885	1.076
6-Gram	1.859	1.936
5-Gram	2.140	2.194

margin. IPMP2 is the best model, however the differences between IPMP2, BOMP and BOMP2 are small. Experiments with Speech Dasher's lattice-based language model are conducted with three different N-gram models on both of the test data sets. Since the PPM model of Speech Dasher is trained on 300 KB of text data, a 5-gram model with similar training data size is used in one of the experiments. Moreover, the 5-gram model used in CATT language model experiments and the 6-gram model trained on 73.9 MB text data are experimented with to see how the performance varies with respect to training data size. The results of SD language model experiments conducted with 5-gram model trained on 4.6 MB text data can be compared with the CATT language models because the N-gram language models are identical for these experiments. Differences in results are accounted for by lattice-based models.

CATT versions of backoff and interpolation schemes perform worse than the ones

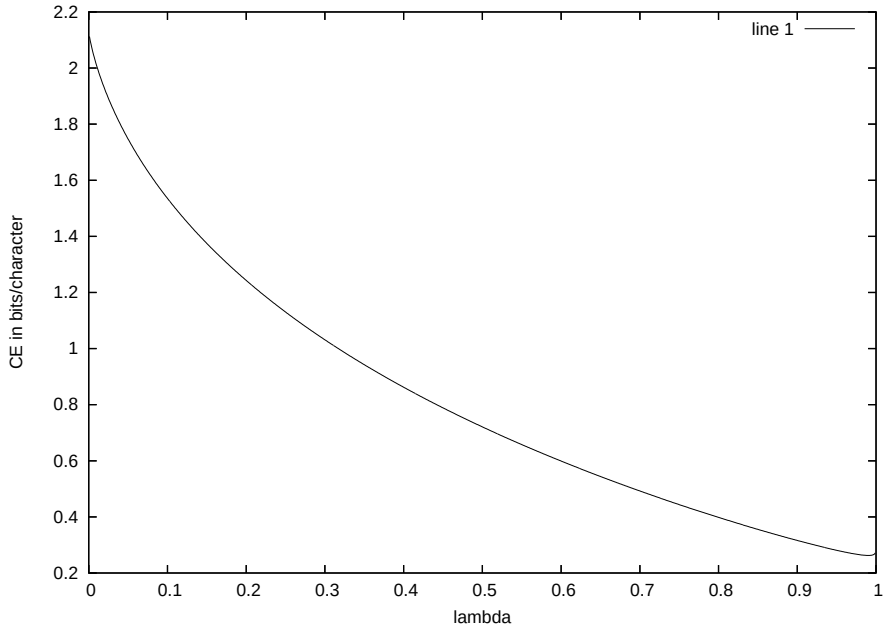

Figure 4.7. CE versus λ for IPMP2 on Test Set 1

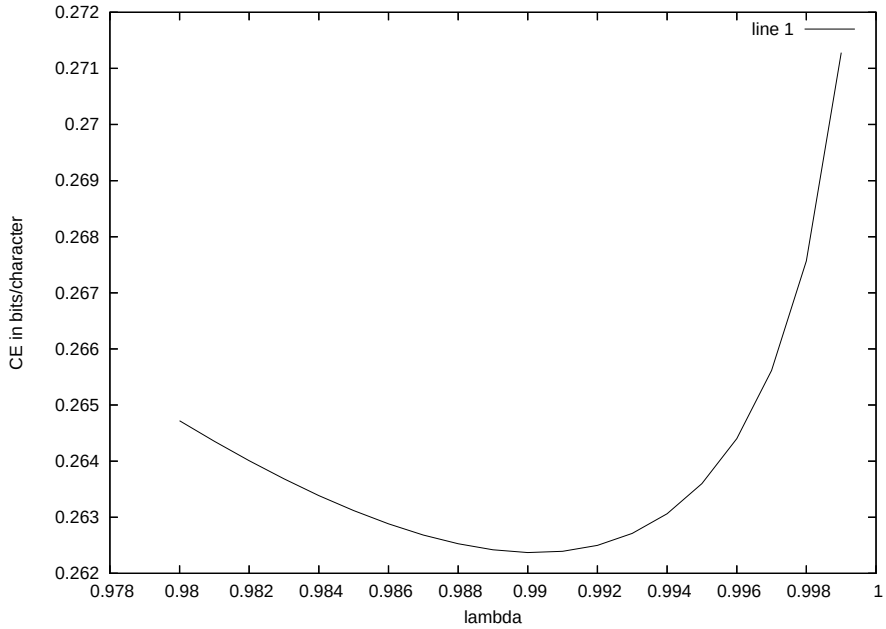

Figure 4.8. CE versus λ for IPMP2 on Test Set 1 (close-up to minimum)

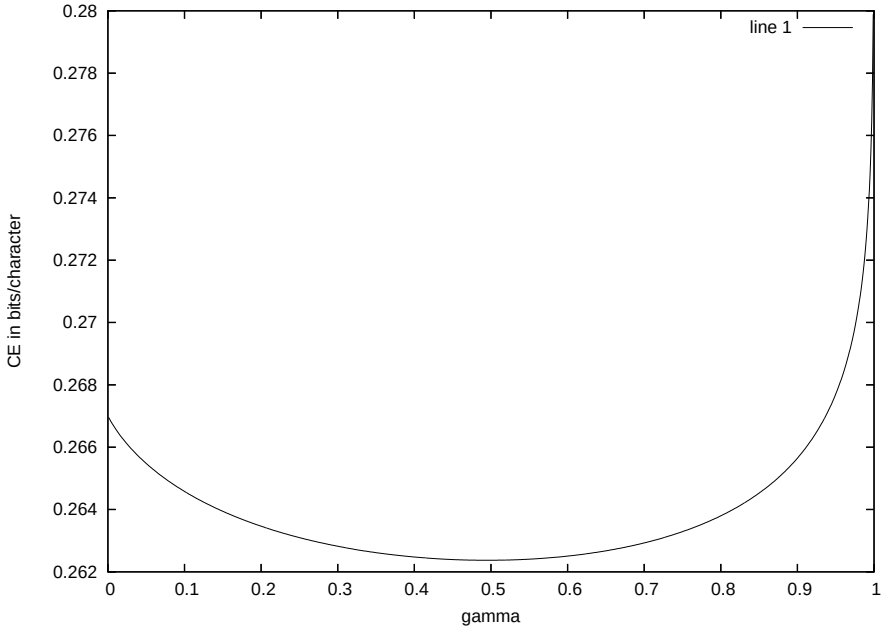

Figure 4.9. CE versus γ for IPMP2 on Test Set 1

looking ahead in the utterance to check for successful lattice expansion. As IPMP2 was better than BOMP and BOMP2, IPCATT performed better than BOCATT. BOMP experiments with test set 2 showed that the optimum value of the c_{DEL} is 0. Hence, BOMP is identical to BOMP2 model for this test set. Since BOGEO model was the worst model on test set 1, no experiments for this model are conducted on test set 2. Since IPMP2 is the best model, the plots of the variation of CE with respect to the model parameters on test set 1 are given for this model in Figures 4.7 through 4.9. Figure 4.8 is a close-up of the Figure 4.7 for the interval $[0.980, 0.999]$.

In Table 4.18, the results of the experiment performed to find out how the models generalize to unseen data are given along with the optimum results obtained on the target data set for easy comparison. As stated above, BOMP model reduces to BOMP2 on test set 2. However, the result for the BOMP model is very close to BOMP2 and, therefore, the optimum result of BOMP2 on test set 2 can be taken as the value to compare with. The same experiment is also performed on test set 1 with parameters optimized on test set 2. The results are given in Table 4.19.

Table 4.18. Results for CATT language model experiments on test set 2 with optimal parameters trained on test set 1

Model	Set 1 parameters on Set 2 CE in bits/char.	Best results on Set 2 CE in bits/char.
BOMP	0.557	-
BOMP2	0.557	0.549
BOCATT	0.598	0.587
IPMP2	0.547	0.539
IPCATT	0.582	0.567

Table 4.19. Results for CATT language model experiments on test set 1 with optimal parameters trained on test set 2

Model	Set 2 parameters on Set 1 CE in bits/char.	Best results on Set 1 CE in bits/char.
BOMP2	0.274	0.268
BOCATT	0.312	0.305
IPMP2	0.268	0.262
IPCATT	0.303	0.294

4.7. Discussion

Various experiments are conducted on letter-based language modeling. In this section, the results of these experiments will be briefly discussed.

The experiments conducted to select the type of the N-gram model and the smoothing technique reveal that all of the models perform very closely. The best result obtained by the KN-BO pair improves the worst result by the DI only by 1%. Any of the models could be used in the combined model and the performance of the combined model would be changed by less than 1% because both of the combination schemes emphasize the lattice, thus the performance difference between models would be scaled down.

N-gram model order experiments are conducted with the same training and test data for all orders. As expected, the higher order models perform better. However, as shown in Figure 4.1, the marginal improvements in cross entropy start to diminish after order 4. This may due to the fact that with increasing model order, the N-gram model starts to act like a word-based bigram model. Proper Turkish syllables contain at most 4 letters [17], and an average word in the text corpus has more than 6 letters. With context lengths more than 4, the number of N-grams that span over 2 words becomes greater than N-grams that are confined to letters of a single word. For model order less than 4, the N-grams reflect the short range letter dependencies.

Increasing the order of the model incurs a model size cost. CATT is not to be run on a high-end work station or a server but on common desktop computers with modest resources because transcribing hundreds of hours of speech data requires multiple transcribers working in parallel. Hence, an application which is intended to be run on such a system should be designed to use as low system resources as possible. The model size with respect to the footprint of the model in the computer memory is an interesting issue to be investigated. Figure 4.2 shows the increase in the number of nodes (states) in the FSM representation of the models. The relative increase in the number of states decreases with increasing model order.

Increasing the model order from 5 to 6 doesn't bring the same amount of decrease in cross entropy as changing from 4-gram to 5-gram model. Moreover, the footprint of the 6-gram on a regular PC may hinder smooth operation of the system. From a theoretical point of view, since the higher the model order is the better the model performance, the highest possible model order would be selected. However, due to practical considerations taken into account, 5-gram language model is used in the combined model experiments. As stated before, the combined model uses the N-gram model as a secondary model, hence the overall performance depends mainly on the properties of the lattice.

The small size of the lexicon in letter N-gram modeling enables the use of less training data than word-based N-gram modeling. How the model performance varies with training data size is also investigated. Although not implemented in this thesis, an effective training data size, determined through such an analysis, may be used as a reference point in weighting the new in-domain data for an adaptive model with respect to the base model [18]. As expected, cross entropy of the test data per character decreases with increasing training data, however, the marginal gain in cross entropy diminishes steadily. This result is in accordance with a similar analysis done on word-based N-gram models [16]. The models become somewhat saturated after training data sizes with more than 500K sentences ($\sim$45.7M characters). Training data set 5 is selected as a compromise between data size and model performance.

The experimental results show that letter level combination of the ASR output with a letter N-gram language model performs better than the 6-gram model and sentence level combination of lattices and N-gram models. If only the lattices were used as language models, then, in case test data set 1, about 30% of utterances couldn't be transcribed since those lattices don't contain the correct utterance as one of the hypotheses. BOCATT model performs worse than other models as expected because misalignment during lattice expansion operations are possible.

Although interpolation method seems to perform better than backoff, it is not suitable for estimating its parameters based on simple statistics derived from the ex-

periments. Although there are methods like expectation-maximization which may be used to estimate model parameters of the interpolation model, these are more expensive than dividing two integers in case of the backoff model. Backoff model includes points where the lattice and the N-gram are alternatives to each other. The parameters of the model reflect how this trade-off is optimized. For example in case of the BOMP model, the optimum value of the parameter modeling the failure event was experimentally found to be 0.996 on test data set 1. The ratio of failure events to total number of characters aligned with the lattice is 0.9959. Backoff method enables online updating of these estimates, and, thus, can be be used as an adaptive model. Experiments conducted with test data set 2 supports this conclusion. In case of BOMP2 model, $\lambda_{\text{Optimal}} = 0.987$ and $\lambda_{\text{Estimated}} = 0.9871$.

The results of the experiments with BOMP and BOCATT models enables us to investigate how the lookahead approach described in Subsection 3.4.4 is reflected in the experimental results. BOMP applies the lookahead approach. BOCATT applies the original backoff developed in detail in Chapter 2. λ_{BOCATT} (0.994 on test set 1) is less than λ_{BOMP} (0.996). This reflects the fact that there are more nodes in the lattice at which the lattice fails. Since deletion is performed only if the lattice fails, unsuccessful deletion attempts increase the number of lattice failures. The changes in γ and c_{DEL} are more significant. c_{DEL} models the deletion events performed upon lattice failures at word boundaries. Due to unsuccessful deletion attempts, $c_{DEL_{BOCATT}}$ (0.323) is greater than $c_{DEL_{BOMP}}$ (0.110). The same reasoning explains the difference in γ_{BOMP} (0.626) and γ_{BOCATT} (0.675). Due to unsuccessful re-entrance attempts, γ_{BOCATT} is greater than γ_{BOMP}.

Although, results obtained from experiments conducted with the test data set 2 are worse than those from test data set 1, the results are consistent. IPMP2 outperforms all other models. SD language model performs worse than CATT language models but better than SLC and N-gram models.

Statistics generated during the backoff combined model experiments indicate that lattice expansion methods play an important role in the success of the models. In over

90% of the lattice failure events during backoff model experiments on test data set 1 and in about 60% of such events in experiments on test data set 2, lattice expansion methods enabled the combined model to return back to the lattice. Out of 51973 characters in the test data set 1, only 3.7% are covered by the N-gram model.

Another statistic that comes from the backoff combined model experiments on test data set 1 show that 31% of the lattice failure events occur at word boundaries. This means that 69% of these events occur within the current word being aligned with the lattice. A word-based lattice would always fail at word boundaries, and, therefore, the lattice as a source of greater probability values would be underused. In case of SLC, the lattice was completely disregarded when it didn't contain the utterance. So no use of the lattice was made in case of failures.

As seen in Tables 4.18 and 4.19, the models perform only slightly worse than the optimum if applied on unseen data with parameters optimized on a different set of data. Comparing these results with the optimum parameter values obtained from combined model experiments conducted on both test data sets, it is the λ parameter that affects the cross entropy most. The variation in γ across test sets is much greater than the variation in λ. Figures 4.7 and 4.9 show that the cross entropy is relatively insensitive to changes in γ. The reason for this behavior lies in the fact that the event γ models, i.e. the re-entrance event back to the lattice, occurs very rarely when compared with the number of characters covered by the lattice at each of which a failure event may occur. This is exactly what λ models, the probability of successful alignment of a character with the lattice. Therefore it is the lattice character error rate with respect to the combined model type that defines how λ is set.

5. CONCLUSIONS

In this thesis, it was shown that the transcription task may be facilitated by providing a transcriber with the "guesses" of an ASR about the utterance to be transcribed. Complementing the ASR output with a suitable N-gram model enables the transcription of utterances which are not encoded in the ASR output. Lattice expansion methods play an important role in overcoming the misalignment errors. Experiments on letter-based N-gram language models are conducted to investigate how model type, model order and training data size affect their performance. A prototype application using the Dasher application as its graphical user interface has been developed. In the rest of this chapter, some conclusions drawn from the experiments are discussed in greater detail.

Data sparseness is one of the main issues in word-based N-gram modeling. As discussed in Section 4.2, the number of trigrams increase exponentially with the number of the entries in the lexicon. However, this is not the case for letter-based modeling. The size of the lexicon is fixed. In case of CATT, only lower case letters are used, therefore 29 letters and a word boundary symbol constitute the lexicon. The 5-gram language model used in the CATT combined language model experiments is trained with 4,174,047 characters. This figure is in the same order as the number of possible 5-gram contexts (24.3M). Hence, data sparseness is not as important as it is for word-based N-gram modeling.

As the experimental results suggest, letter-based N-gram modeling is relatively insensitive to training data size compared to word-based N-gram modeling. As reported in [16], increasing training data size from 10,000 to 100,000 sentences decreases cross entropy as much as 12% for word-based N-gram models, whereas, in case of letter-based models, the change remains below 6%. Note that the training data used in [16] consists of sentences with, on the average, 25 words, whereas the sentences in the text corpus used in this thesis has, on the average, 12.5 words in the case of test data set 1. If the result is adjusted according to this fact, the change in cross entropy of the letter-based

models remains below 5%.

On the other hand, the performance of the speech recognizer greatly affects the performance of the combined language model. As seen in Table 4.2, the lattices in test data set 2 are worse than those in test data set 1 with respect to lattice utterance error rate. Therefore, the combined model relies more on the N-gram model in case of test data set 2, hence greater cross entropy results in the experiments.

Speech Dasher language model experiments show that using the original output lattices of a speech recognizer gives better results compared to n-best lists which poorly reflect the probability space of the recognizer. However, exploiting recognizer output proves to be useful, if compared with methods not exploiting such information, because, as shown in Table 4.17, Speech Dasher language model outperforms SLC and N-gram models.

Some screenshots of the application prototype, which are taken while a sentence which is contained in the lattice is entered, are given in Appendix A.

APPENDIX A: CATT in Action

Some screenshots of the CATT prototype are given in Figures A.1 through A.3 while entering the sentence "`bin ladin için geri sayım`".

The following figures illustrate the lattice expansion methods. The lattice correponds to the utterance "`yeni kurulan yedi sekiz dağıtım şirketinin çok hızlı bir şekilde depolar kurduğunu kaydeden öztürk yeni dağıtım şirketlerinin denetlenmesini istedi`".

In Figure A.4, the model is at the word boundary after the word `yedi`. Not only the following word `sekiz` but also the word following it, `dağıtım`, is shown large enough so that a deletion opertaion can be easily performed.

In Figure A.5, the model is at the word boundary after the word `su` which is inserted between `sekiz` and `dağıtım`. As seen in the figure, it is fairly easy to return to lattice and enter `dağıtım`.

In Figure A.6, the model is at the word boundary after the word `tane`. However this word is not in the lattice, it has substituted the word `sekiz`. The word following `sekiz` in the utterance is `dağıtım`, and it is one of the words that can be easily entered at the given context after substitutuion.

The Figure A.7 illustrates the case in which a failure occurs while entering the word `kur`***`duğunu`***. The model leaves the lattice and the rest of the word `kur`***`ulacak`*** is covered by the N-gram model. After this is entered, CATT presents the words that are more likely to be entered. As seen in the figure, the word `yedi`, which is the word following `kurduğunu` in the utterance, can be easily entered.

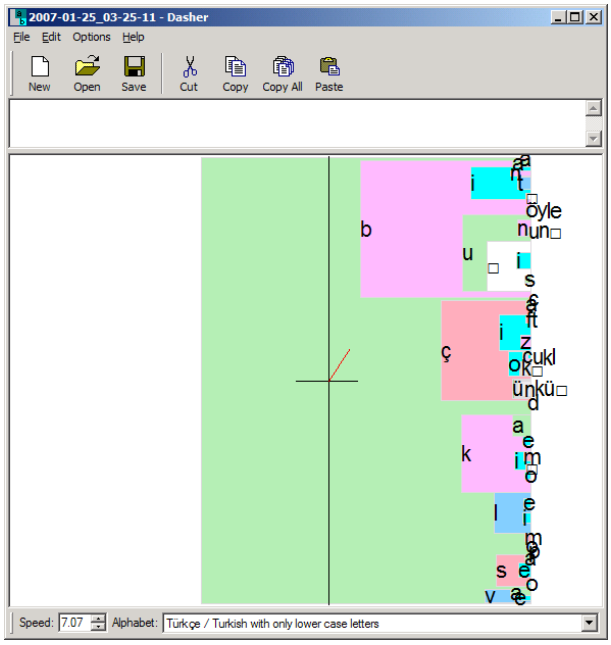

Figure A.1. Screenshot of CATT - beginning of sentence

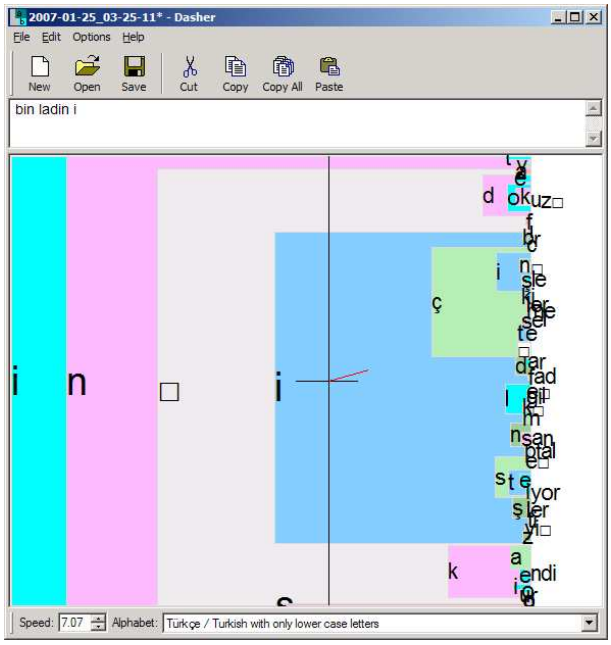

Figure A.2. Screenshot of CATT - middle of sentence

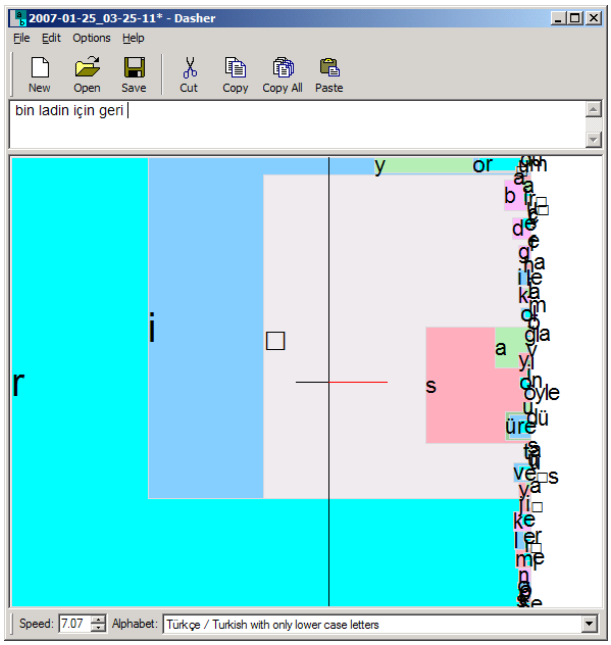

Figure A.3. Screenshot of CATT - at the last word of sentence

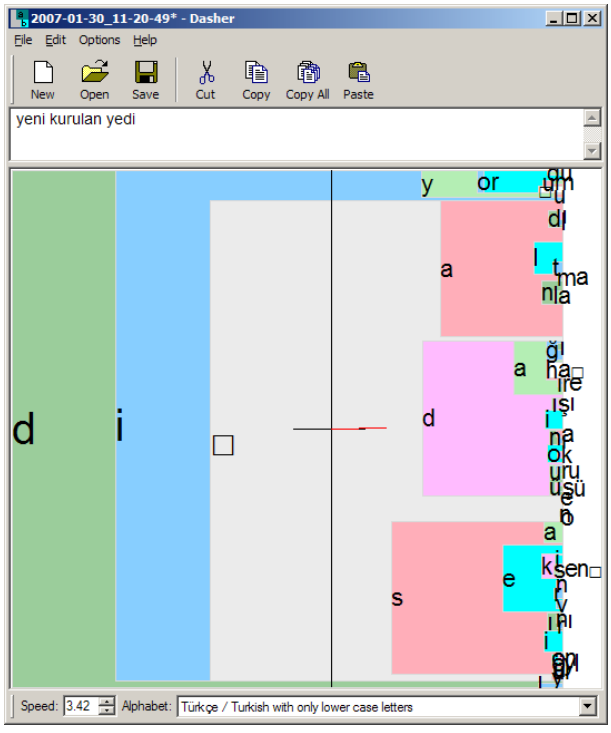

Figure A.4. Screenshot of CATT - Deletion

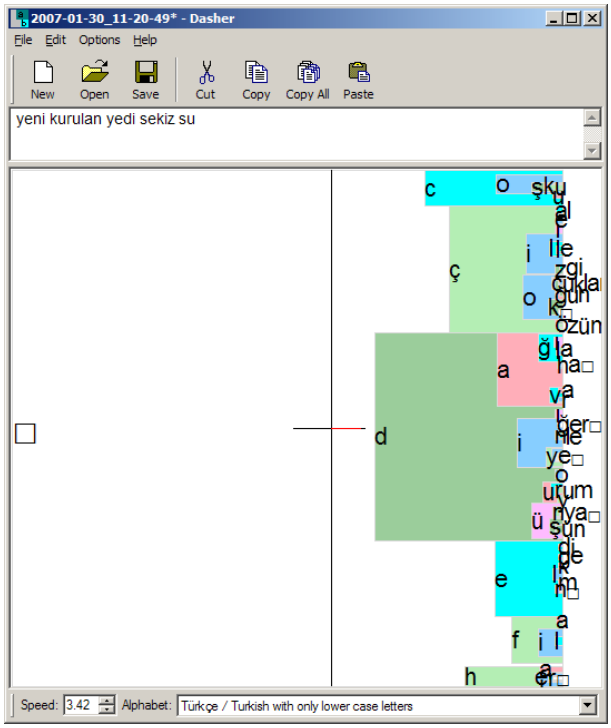

Figure A.5. Screenshot of CATT - Insertion

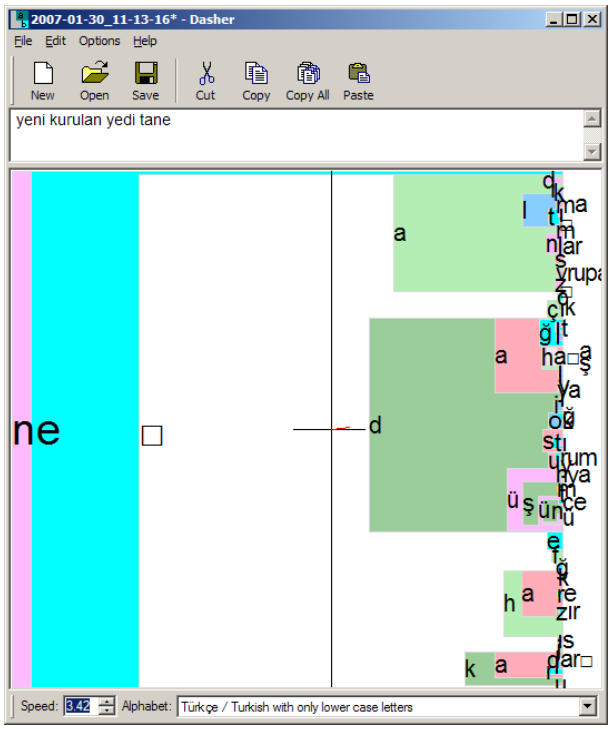

Figure A.6. Screenshot of CATT - Substitution

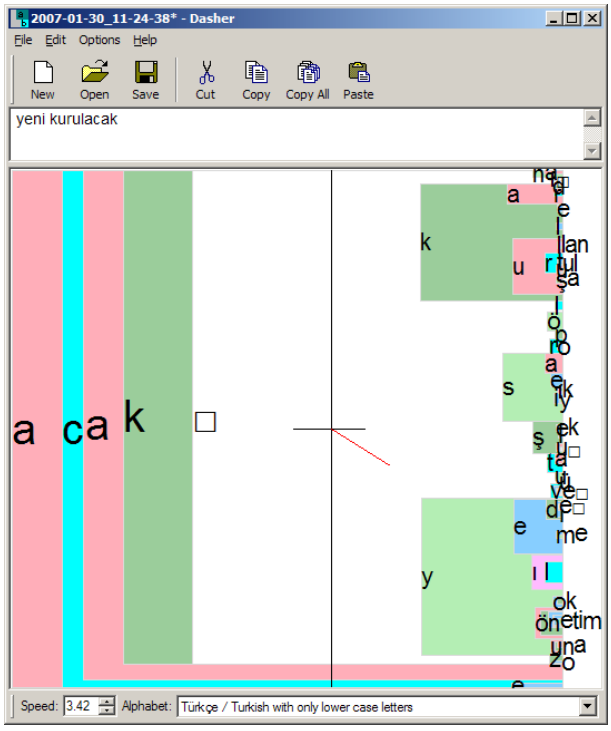

Figure A.7. Screenshot of CATT - Back to lattice after leaving it within a word

REFERENCES

1. Ward, D. J., A. F. Blackwell, and D. J. C. MacKay, Dasher - A Data Entry Interface Using Continuous Gestures and Language Models, *Proceedings of UIST 2000*, pp 129–137, 2000.

2. Ward, D. J., 2001, Adaptive Computer Interfaces, *PhD Thesis*, University of Cambridge, 2001.

3. The Dasher Project, http://www.inference.phy.cam.ac.uk/dasher/.

4. Moffat, A., Implementing the PPM Data Compression Scheme, *IEEE Transactions on Communications*, Vol. 38, No. 11,pp 1917–1921, 1990.

5. Witten, I. H., A. Moffat and T. C. Bell, *Managing Gigabytes - Compressing and Indexing Documents and Images*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1999.

6. Witten, I. H. and T. C. Bell, The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression, *IEEE Transactions on Information Theory*, 37(4), pp 1085–1094, 1991.

7. Vertanen, K., Efficient Computer Interfaces Using Continuous Gestures, Language Models, and Speech, *M.Phil Thesis*, University of Cambridge, 2004.

8. Jelinek, F., *Statistical Methods for Speech Recognition*, MIT Press, Massachusetts, 1997.

9. Huang, X., A. Acero and H. W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Devolopment*, Upper Saddle River, NJ, 2001.

10. Jurafsky, D. and J. H. Martin, *Speech and Language Processing*, Upper Saddle River, New Jersey, 2000.

11. Hakkani-Tür, D. Z., K. Oflazer and G. Tür, Statistical Morphological Disambiguation for Agglutinative Languages, *In Proceedings of COLING 2000*, ICCL, 2000.

12. Mohri, M., F. Pereira and M. Riley, Weighted Automata in Text and Speech Processing, *ECAI 96 12th European Conference on Artificial Intelligence*, 1996.

13. Mohri, M., Weighted Grammar Tools: The GRM Library, in *Robustness in Language and Speech Technology*, Jean-Claude Junqua and Gertjan van Noord (eds), Kluwer Academic Publishers, pp 19–40, 2000.

14. AT&T FSM Library, http://www.research.att.com/∼fsmtools/fsm/.

15. Mohri, M., Finite-State Transducers in Language and Speech Processing, *Computational Linguistics*, Vol. 23, No.2, 1997.

16. Chen, S. F. and J. Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, *Proceedings of the 34th Annual Meeting of the ACL*, June 1996.

17. Gönenç, G., Unique Decipherability of Codes with Constraints with Application to Syllabification of Turkish Words, *Computational Linguistics*, 1973.

18. Bellegarda, J. R., Statistical Language Model Adaptation: Review and Perspectives, *Speech Communication*, 42, pp 93-108, 2004.