Preface

This work was done in the Laboratory of Computer and Information Science of Helsinki University of Technology during years 2004 and 2005. This study was facilitated by the huge work done in the speech recognition group and I am grateful for the opportunity to work in it. I thank professor Timo Honkela for supervising my work. I thank my instructor Mikko Kurimo for suggesting this interesting topic and for the valuable corrections. Special thanks to the whole speech recognition group for the interesting conversations and good atmosphere. I also thank Mathias Creutz for providing the morph dictionary used in this work. Finally, I wish to thank Hanna and my whole family for support and encourage.

Simo Broman
Otaniemi, April 4, 2005

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä: Simo Broman
Työn nimi: Combination Methods for Language Models in Speech Recognition
Suomenkielinen nimi: Kielimallien yhdistämismenetelmiä puheentunnistuksessa

Päivämäärä: 4. huhtikuuta 2005	**Sivumäärä:** 64
Osasto: Teknillinen fysiikka ja matematiikka **Professuuri:** Informaatiotekniikka	
Valvoja: Prof. Timo Honkela	**Ohjaaja:** Dosentti Mikko Kurimo

Tilastollinen kielimalli on oleellinen osa nykyaikaista puheentunnistusjärjestelmää, jossa sen tehtävä on pisteyttää sanahypoteesit kielellisen informaation perusteella. Lukuisia kielimalleja on esitetty kirjallisuudessa. Parhaat tulokset on saavutettu käyttämällä eri kielimalleja yhdessä. Useita menetelmiä kielimallien yhdistelyyn on esitetty, mutta kattavaa tutkimusta eri menetelmistä ei ole esitetty.

Tässä työssä tutkitaan kirjallisuudessa esitettyjä yhdistämismenetelmiä. Lisäksi työssä esitetään uusi menetelmä, joka perustuu uskottavuustiheysfunktion estimointiin histogrammien avulla. Teoreettisen tarkastelun lisäksi neljää yhdistämismenetelmää arvioidaan puheentunnistuskokeilla sekä kielimallin hyvyyttä kuvaavilla perplexity-kokeilla. Aineistona käytetään suomenkielisiä uutisartikkeleita. Yhdisteltävinä kielimalleina toimii neljä kielimallia, jotka esitellään työssä.

Perplexity-kokeissa kaikilla yhdistämismenetelmillä saavutettiin kielimalleista riippuen tilastollisesti merkittävää parannusta vertailukohtana toimineeseen 4-grammi-malliin. Paras tulos, 46 % parannus 4-grammimalliin verrattuna, saatiin yhdistämällä useita malleja uudella bin-estimointi-menetelmällä. Puheentunnistuskokeissa saavutettiin parhaimmillaan 4 % parannus sanavirheessä ja 7 % parannus äännevirheessä.

Avainsanat: kielenmallinnus, puheentunnistus, yhdistämismenetelmät, kielimallien yhdistäminen

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author: Simo Broman
Title: Combination Methods for Language Models in Speech Recognition
Title in Finnish: Kielimallien yhdistämismenetelmiä puheentunnistuksessa

Date: April 4, 2005 | **Pages:** 64

Department: Engineering Physics and Mathematics
Professorship: Computer and Information Science

Supervisor: Prof. Timo Honkela | **Instructor:** Docent Mikko Kurimo

Statistical language models have a vital part in contemporary speech recognition systems, where the purpose of the language model is to score the word hypothesises based on the linguistical knowledge. A lot of language models have been presented in the literature. The best results have been achieved when different language models have been used together. Several combination methods have been presented, but a thorough investigation of the methods has not been done.

In this work, combination methods that have been used with language models are studied. Also, a new approach based on likelihood density function estimation using histograms is presented. In addition to theoretical consideration, four combination methods are evaluated in speech recognition experiments and perplexity calculations that measure the quality of the language models. The test data consist of Finnish news articles. Four language models, that are presented in the work, work as the component models.

In the perplexity experiments, all combination methods produced statistically significant improvement compared to the 4-gram model that worked as a baseline. The best result, 46 % improvement to the 4-gram model, was achieved when combining several language models together by using the new bin estimation method. In the speech recognition experiments, 4 % reduction to the word error and 7 % reduction to the phoneme error was achieved.

Keywords: language modeling, speech recognition, combination methods, combining language models

Contents

Symbols and abbreviations

$c(X)$	Number of occurrences of X
$E[X]$	Expectation of X
$P(X)$	Probability of X
$p(X)$	Likelihood of X
$\mathcal{V}$	Vocabulary
w_i^j	Word sequence $w_i, w_{i+1}, ..., w_j$

LSA	Latent semantic analysis
ME	Maximum entropy
ML	Maximum likelihood
OOV	Out-of-vocabulary
PHER	Phoneme error rate
WER	Word error rate

Chapter 1

Introduction

1.1 Introduction

Language modeling has a vital part in contemporary speech recognition systems and many other areas of language technology including character recognition, machine translation, and spelling correction. The target of the language modeling is to capture linguistical or statistical information of the language and present it in a useful form.

Language modeling can be divided into linguistical and statistical language modeling. The linguistical approach is more intuitive as it gives an answer to the question, whether a sentence or a word sequence is legitimate or not. For automatic speech recognition this is however insufficient. Statistical language modeling focuses on collecting information of statistical relationships between words. In speech recognition, the statistical approach is more useful, as the language model is needed to determine the probability of how likely it is a word or a word sequence to occur given the word history.

The most popular language modeling paradigm is the family of n-gram models. Though simple, n-gram models have proven to be powerful and hard to outperform. Lot of work has been done in developing models that would better utilize syntactic or semantic structure of the language. In literature, language models that model different aspects of language have successfully been combined together. A lot of methods for combining language models together have been presented. However, a thorough investigation of different combination methods has not been done.

The purpose of this work is to study different methods that have been used in combining language models. The objective is to compare the pros and cons of the methods and to form a picture in which techniques are applicable in different situations. Also

a new approach based on multivariate function estimation is presented for combining language models.

In the experimental phase of the work, four combination methods, linear interpolation, log-linear interpolation, unigram rescaling, and the new bin estimation method, are used in combining four language models. The experiments are run on Finnish news data and the methods are evaluated using both perplexity experiments and speech recognition tests. Most of the experiments are reproductions of works presented in the literature. However, the Finnish language and the use of morpheme-like sub-word units as the basic units bring in a fresh aspect.

The organization of the work is following. In the second part of the introduction (section 1.2) special properties of the Finnish language are considered from the point of view of language modeling and speech recognition. In chapter 2 the language models that are used in the experiments are represented. In chapter 3 the combination methods that are used in the language modeling are discussed. In chapter 4 the experimental setup is explained and the results are analyzed. In the last chapter (5) the final conclusions are summed up.

1.2 Properties of Finnish language

Finnish language has properties that make it more challenging target for language modeling and speech recognition than for example English. Finnish has a big number of inflectional forms resulting in a big number of affixes that can be combined successively, and also the word stems may be affected. This means that the total number of possible words is enormous and traditional vocabulary based approaches can not be used in continuous speech recognition. Even calculating only the base forms, Finnish has quite a large vocabulary further emphasizing the problem.

Great deal of speech recognition research is done for English. Many of the developed methods are universal and widely used for other languages, too. However, many languages have features that produce problems that do not occur in English.

In Finnish, long and short phonemes are distinguished. Pylkkönen (2004) has studied modeling of the phone duration in the hidden Markov model framework and shows that the duration modeling improves the speech recognition results for Finnish. Phone durations do not, however, affect language modeling and the subject is not considered in this work.

Because information about the relations of the sentence constituents is given in inflectional affixes, the word order in a sentence is quite relaxed. This may also degrade the performance of the n-gram models that rely on the word order.

From the point of view of statistical language modeling, the curse of dimensionality caused by the large vocabulary and inflection is even worse problem in Finnish than in many other languages. Some approaches to alleviate the problem have been suggested. A natural approach is to use base forms of words. The problem with the base forms is that the probability estimates must be determined for the inflected forms, too. Base forms have been used for example in capturing the semantic content of documents by Kurimo and Lagus (2002).

Another approach is to make use of the morphological structure of words (Geutner, 1995), (Schultz et al., 1999). Words can be decomposed to morphemes that are considered to be the smallest meaning-bearing units of language. Creutz and Lagus (2002) have presented two algorithms for automatically deriving morpheme-like units, 'morphs', that try to approximate the natural morphemes. These morphs have been used in automatic speech recognition by Siivola et al. (2003) resulting significantly better performance than by using whole words or syllables. This approach is also used in this work. The automatically derived morpheme-like units are learned from the training data and used as the basic units in all language models.

Chapter 2

Language models

2.1 N-gram models

N-gram models are the most important language models and standard components in contemporary speech recognition systems. N-gram model is a simple but efficient predictor for the following word given the immediately preceding words. The term n-gram means a word sequence of length n. The n-gram models are simple Markov models that make the assumption that the probability of the following word depends only on the $n-1$ preceding words. N-gram models are trained with a text corpus and the probability is estimated using the maximum likelihood estimate

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} \tag{2.1}$$

where w_{i-n+1}^{i-1} refers to word sequence $w_{i-1}w_{i-2}\dots w_{i-n+1}$ and $c(w_{i-n+1}^{i})$ is the number of times the word sequence w_{i-n+1}^{i} occurs in the data. The simplest n-gram model, having $n = 1$, is just the word probability regardless of the context. When the value of n is increased, longer word history is taken into account and better discrimination between words can be made. However, the complexity of the model grows exponentially as the function of the order n. Thus, the greater order the more data is needed to train the model and more memory is needed to store the model. These reasons limit the model order. Typically the models used in speech recognition systems are of order 2 to 5.

Higher order n-grams make better discrimination between words, while lower order n-grams are more robustly trained with less training data. To take advantage of the both, several n-gram models of different orders are trained and combined together and used as a single model. Usually the combining is done using *backoff* (Katz, 1987) or *linear interpolation*. Also maximum entropy modeling has been used by Rosenfeld

(1994). Usually the term n-gram model is used to refer to a compound model built from the models of order 1 to n.

Smoothing

With a vocabulary of size $|\mathcal{V}|$, the number of possible n-grams is $|\mathcal{V}|^n$. The typical size of the vocabulary is tens of thousands of words. When $n > 1$ the number of the n-grams becomes large. Of course, most of these are extremely rear. However, training a model that has such amount of parameters with data of any practical size is prone to overfitting. Maximum likelihood estimate gives systematically too large probabilities for events that have been seen in the training data and too small probabilities, i.e. zero, for unseen ones. In speech recognition, a word having a zero probability will automatically produce an error. In perplexity calculations the zero probability makes the perplexity to go infinite. Thus, for every word in the vocabulary we want a probability greater than zero in every context. This is done so that for each word that has been seen in the data, the probability is lowered from its ML-estimate and the probability mass is distributed to the unseen words. The procedure is called smoothing since the resulting distribution is more uniform or smoother than the original distribution.

Several methods have been proposed to carry out the smoothing of the probability distributions. These include Good-Turing discounting (Good, 1953), Katz smoothing (Katz, 1987), Witten-Bell discounting (Witten and Bell, 1991), and Kneser-Ney smoothing (Kneser and Ney, 1995). Smoothing has remarkable effect on the performance of the model. The experiments of Chen and Goodman (1998) show that Kneser-Ney and its modification outperform other smoothing methods. The Kneser-Ney smoothing is also used with the n-gram models used in this work. In this section, the main ideas of smoothing are presented. For more information about smoothing methods, see the extensive study of Chen and Goodman (1998).

Smoothing methods consider three things: discounting, construction of the lower order distributions, and combination of the different order n-grams. In discounting, the count of how many times an event is observed in the data is modified. For the events that are seen in the data, the count is lowered from the observed count. The combination of the different order n-gram models is usually done using backoff or linear interpolation. In most smoothing methods the lower order distributions are constructed from the discounted counts like the highest order distribution. In the Kneser-Ney smoothing the lower order distributions are calculated in a different way based on the number of different contexts the particular word occurs after.

Most existing smoothing algorithms can be described with recursive equation

$$p_{smooth}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) p_{smooth}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \quad (2.2)$$

where $\alpha(w|h)$ is the distribution calculated from the discounted counts, $\gamma_{w_{i-n+1}^{i-1}}$ is a backoff term that is chosen to make the conditional probabilities sum to one, and $c(w_{i-n+1}^{i})$ is the count how many times word sequence w_{i-n+1}^{i} has occurred in the training data. If an n-gram has a nonzero count then the distribution $\alpha(w_i|w_{i-n+1}^{i-1})$ is used. Otherwise, we backoff to the lower-order distribution $p_{smooth}(w_i|w_{i-n+2}^{i-1})$. Chen and Goodman (1998) call methods that fall directly into this framework as *backoff* models. (Chen and Goodman, 1998)

Instead of using backoff, lower order n-grams can be combined with higher order n-grams using linear interpolation:

$$p_{smooth}(w_i|w_{i-n+1}^{i-1}) = \lambda_0 p_{ML}(w_i) + \sum_{j=1}^{n-1} \lambda_j p_{ML}(w_i|w_{i-j}^{i-1}) \quad (2.3)$$

where $\lambda_j \in [0, 1]$ and $\sum_{j=0}^{n-1} \lambda_j = 1$. Sometimes this is further interpolated with the uniform distribution

$$p_{uniform}(w_i) = \frac{1}{|\mathcal{V}|} \quad (2.4)$$

where $|\mathcal{V}|$ is the size of the vocabulary. Using the recursive notation as in equation 2.2 the interpolated models can be described by equation

$$p_{smooth}(w_i|w_{i-n+1}^{i-1}) = \lambda_{n-1} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{n-2}) p_{smooth}(w_i|w_{i-n+2}^{i-1}) \quad (2.5)$$

This can be rewritten as

$$p_{smooth}(w_i|w_{i-n+1}^{i-1}) = \alpha'(w_i|w_{i-n+1}^{i-1}) + \gamma_{w_{i-n+1}^{i-1}} p_{smooth}(w_i|w_{i-n+2}^{i-1}) \quad (2.6)$$

where

$$\alpha'(w_i|w_{i-n+1}^{i-1}) = \lambda_{n-1} p_{ML}(w_i|w_{i-n+1}^{i-1}) \quad (2.7)$$

and

$$\gamma_{w_{i-n+1}^{i-1}} = 1 - \lambda_{n-1}. \quad (2.8)$$

By taking

$$\alpha(w_i|w_{i-n+1}^{i-1}) = \alpha'(w_i|w_{i-n+1}^{i-1}) + \gamma_{w_{i-n+1}^{i-1}} p_{smooth}(w_i|w_{i-n+2}^{i-1}), \quad (2.9)$$

it is seen that also these models can be presented by equation 2.2. Chen and Goodman (1998) call this kind of models as *interpolated* models. They show that interpolated models generally perform better than backoff models.

An essential part in the smoothing methods is the discounting. Instead of the count of how many times an n-gram has been observed in the data, a modified value is

used. The purpose of the discounting is to shift probability mass from the observed events to the unseen events. A lot of discounting methods have been presented in the literature. Here two most important ones, Good-Turing and absolute discounting, are described.

Good-Turing discounting (Good, 1953) is central to many smoothing techniques. The Good-Turing discounting means that for any n-gram that occurs r times, modified count r^* is used, calculated as

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{2.10}$$

where n_r is the number of n-grams that occur exactly r times in the training data. This is converted to a probability by normalizing. For an n-gram w_k^{k+n-1} with r counts

$$P_{Good-Turing}(w_k^{k+n-1}) = \frac{r^*}{\sum_{r=0}^{\infty} n_r r^*} \tag{2.11}$$

Rewriting the denominator as

$$\sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty}(r+1)n_{r+1} = \sum_{r=1}^{\infty} r n_r \tag{2.12}$$

we see that the denominator is the original number of counts in the distribution. (Chen and Goodman, 1998)

In the absolute discounting (Ney et al., 1994) a fixed count $D \leq 1$ is subtracted from each nonzero count. Ney et al. (1994) suggest setting D through deleted estimation on the training data arriving at the estimate

$$D = \frac{y_1}{y_1 + 2y_2} \tag{2.13}$$

where y_1 and y_2 are the total number of n-grams with exactly one and two counts. Chen and Goodman (1998) suggest using different discounting terms D_1, D_2, and D_{3+} for cases where the n-gram has occurred 1, 2, or three or more times. In practice, the discounting constants are usually optimized on held out data.

Kneser-Ney smoothing

Kneser-Ney smoothing (Kneser and Ney, 1995) takes advantage of the absolute discounting. In addition, the lower order distributions are built in a novel manner, while in other smoothing methods the discounted lower order distributions are used as such. The lower order n-gram information is useful mainly in cases where the higher order information is missing or estimated only from few counts. So the lower order distributions should be optimized for the use in these cases.

Kneser-Ney smoothed backoff model is described by equation

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{\max\{c(w_{i-n+1}^{i})-D,0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) P_{KN}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \quad (2.14)$$

where $c(w_{i-n+1}^{i-1})$ is the number of times the word sequence w_{i-n+1}^{i-1} has occurred in the data, D is a constant determined by equation 2.13 or optimized on the data, and $\gamma(w_{i-n+1}^{i-1})$ is the backoff term that is chosen to make the probabilities to sum up to one. In Kneser-Ney smoothing the lower order distribution is chosen so that the marginals of the smoothed higher order distribution will match the marginals of the training data

$$\sum_{w_{i-n+1} \in \mathcal{V}} p_{KN}(w_{i-n+1}^{i}) = \frac{c(w_{i-n+2}^{i})}{\sum_{w_i} c(w_{i-n+2}^{i})} \quad (2.15)$$

The left hand side of the equation is the marginal probability for w_{i-n+2}^{i} given the smoothed $(n-1)$-gram distribution p_{KN}. The right-hand side of the equation is the observed frequency of w_i to occur after word sequence w_{i-n+2}^{i-1} in the data. Kneser and Ney (1995) show that this leads to expression

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet)} \quad (2.16)$$

where

$$N_{1+}(\bullet w_{i-n+2}^{i}) = |\{w_{i-n+1} : c(w_{i-n+1}^{i}) > 0\}| \quad \text{and} \quad (2.17)$$

$$N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet) = |\{(w_{i-n+1}, w_i) : c(w_{i-n+1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^{i}). \quad (2.18)$$

In words, $N_{1+}(\bullet w_{i-n+2}^{i})$ refers to the number of different words that precede the $(n-1)$-gram w_{i-n+2}^{i}. For the derivation, see (Kneser and Ney, 1995) or (Chen and Goodman, 1998). This means that the lower order distribution is built from the number of different contexts the word or the word sequence occurs in the data. Goodman (2000) gives a theoretical justification for building the lower order distributions so that they satisfy the equation 2.15

Variations

Since their introduce a lot of variations have been suggested to the basic n-gram paradigm. As one moves towards higher order n-grams, the chance of having seen the exact context in the training data becomes very small. Instead, contexts where most of the words are same may have been seen. To utilise this this observation, skipping models have been used by Rosenfeld (1994), Huang et al. (1993), Ney et al. (1994), Martin et al. (1999), Siu and Ostendorf (2000). In skipping models, one or

more words in the context are skipped. For example, a model is created to estimate probability $P(w_i|w_{i-1}w_{i-3}w_{i-4})$ or $P(w_i|w_{i-2}w_{i-3}w_{i-4})$. (Goodman, 2001)

Another more commonly used approach to ease the problem of data sparsity is word clustering. Word clustering can be used with many kind of language models, not only n-gram models, and the basis on how the clustering is done varies by the purpose. With n-gram models the purpose of clustering is to group words that occur in similar contexts together. For example, we may have seen the sentence *"A man bought some bread"*. If we have managed to cluster words *bread* and *cake* to the same cluster, we might be able to give the sentence *"A man bought some cake"* high probability even if the word *cake* had never occurred in the particular context. Different clustering may be used for words in the context and words to be predicted. Besides improving perplexity, clustering may also help to compress the language model. (Goodman, 2001)

Though simple, n-gram models have shown to be efficient and hard to beat. Usually new language models that are introduced incorporate also an n-gram model in it. The reasons for the use of the n-gram model are its efficiency and simplicity and good performance.

2.2 Modeling long range dependencies

The n-gram models have some well known drawbacks. While the n-gram models are good in capturing the local dependence of words, they do not take any long range information like semantic relationships of the words into account. In language, the vocabulary may vary strongly depending on the subject and the situation. While the n-gram models are good in capturing the local dependencies of words, they fail to model long range dependencies, like topical coherence, between the words.

Several approaches have been proposed to utilise long span information in language modeling. The most simple method is cache model, first presented by Kuhn and de Mori (1990), that raises the probability of the words that have occurred in the history. Generalisation to this are trigger models (Rosenfeld, 1994) in which an occurrence of a word raises also the probabilities of some other words that occur often in the same context.

In mixture models (Iyer and Ostendorf, 1999) the text is segmented into units that can be sentences, paragraphs, or articles. Similar units are clustered together and for each cluster a separate n-gram model is trained. The probability of a word is

then calculated as a weighted sum of each model as

$$P(w_i|h) = \sum_{k=1}^{K} \lambda_k P_k(w_i|h). \tag{2.19}$$

where again $\lambda_k \in [0,1]$, $\sum_{k=1}^{K} \lambda_k = 1$, and P_k is the probability estimate given by the k'th model. Different similarity measures can be used for clustering. Iyer and Ostendorf use a measure based on the number of content words in common between the two clusters. Regardless of the exact form of the similarity measure, the clusters are assumed to be homogeneous in some sense, and better models are expected to be trained for these coherent clusters.

Kurimo and Lagus (2002) use self organizing map in clustering documents based on their word distributions. Specific bigram models are built for each cluster as in mixture models. Chen et al. (1998b) propose several methods for topic adaptation using unnormalized exponential models, while Martin et al. (1997) use variable length n-grams to achieve topic dependency.

In this work, three models that try to catch long range dependencies are used: cache model, LSA model, and topic model. A bigram cache model stores all words and bigrams that have been encountered in the document so far. A model based on latent semantic analysis (Bellegarda, 2000), referred here as the LSA model, models the semantic relations between words based on their co-occurrences in the documents. The third model is the topic-based language model presented by Gildea and Hofmann (1999). In the following chapters, these models are represented in the form they are used in this work.

2.3 Cache model

A word that has occurred in the past is much more likely to re-occur in the near future than would be expected from its global frequency. A simple way to utilize this phenomenon in language modeling is to keep track of the words that have occurred and to raise their probability estimates in the future. This kind of models are known as cache models, first introduced by Kuhn and de Mori (1990) and Jelinek et al. (1991)

A simple approach to implement a cache model is to store the words that have been covered so far to the cache and keep count of their occurrences. A generalization to this is to store n-grams with $n > 1$ in addition to the unigrams. An ordinary n-gram model is constructed from the content of the cache, and the probability is determined

as

$$P_{unicache}(w_i) = \frac{1}{i-1}\sum_{j=1}^{i-1} I(w_j = w_i) \tag{2.20}$$

where $I(x)$ is an indicator function which is 1 if x is true and 0 otherwise. The generalization to higher order n-grams is presented as

$$P_{ngram\ cache}(w_i|w_1^{i-1}) = \frac{\sum_{j=1}^{i-n} I(w_j^{j+n-1} = w_{i-n+1}^{i})}{\sum_{j=1}^{i-n} I(w_j^{j+n-2} = w_{i-n+1}^{i-1})} \tag{2.21}$$

The different order n-grams can be combined using backoff or linear interpolation. We choose to use the interpolation, because backoff would need recalculation of the backoff weights each time a word is appended to the cache. As the cache model is constructed based on very little data, the higher order n-grams are very unreliable and the lower order information is more useful than in normal n-gram models. To make the best use of the different order n-grams, the interpolation weights are adjusted depending on the frequency of the context. The context means here the $n-1$ previous words where n is the order of the highest order n-gram used in the cache model. If the context has occurred several times in the history, more weight is given to the higher order n-gram, and if the context has not occurred before, the cache probability is purely determined by the unigram frequencies. Adjusting the interpolation weights based on the frequency of the context has been studied by Jelinek and Mercer (1980).

In this work, a bigram cache is used, i.e. unigrams and bigrams are stored into the cache. The probability is determined as a weighted sum of unigram and bigram probabilities as $P(w_i|w_{i-1}) = \beta P_{uni}(w_i)+(1-\beta)P_{bi}(w_i|w_{i-1})$ where the interpolation weight β is adjusted by formula

$$\beta(w_{i-1}) = \max\{\beta_0(1 - \frac{c(w_{i-1})}{a}), b\} \tag{2.22}$$

where constants β_0, a, and b, for which $0 \leq \beta_0 \leq 1$, $a > 0$, and $0 \leq b \leq 1$, are optimised on the development data. Term a determines the sharpness of the slope and term b sets a minimum value under which the weight of the unigram cache will not fall.

Discounting could be used as is normally done with the n-grams models. This is however irrelevant, specially when not using the backoff method, since the purpose of the cache is just to boost the probabilities of the words that have occurred in the near history, and not to try to construct a language model for all words. Cache model is always used together with other models like an n-gram model.

Clarkson and Robinson (1997) has shown that the probability of a word to reoccur decreases as a function of the distance to the previous occurrence. One way to take advantage of this information is to limit the cache size. Clarkson used successfully

a model where the probability estimate of a word was exponentially decreasing as a function of the distance to the last occurrence of the word. If the data in use has been segmented to topically homogeneous documents it may also be useful to clear the cache between the documents.

The trigger model presented by Rosenfeld (1994) can be considered as a generalization to the cache model. In trigger model an occurrence of a word increases the probabilities of also some other words than just the word itself. Rosenfeld uses maximum entropy framework in combining the trigger information to the n-gram model.

Three-value cache

It is not clear what is the optimal way to formulate the cache model and how to turn the information in the cache model into a probability estimate. The cache model presented above is a rather heuristic way of boosting the probabilities of the recently occurred words. In this subsection I will present another formulation of the cache model that we call three-value cache. The structure of the model is similar to the model presented above but the difference is in how the output is defined. The three-value cache gives as the output one of the three values: 0, 1, and 2. The output is

0 if the word is not in the cache
1 if the word is in the cache but not with the current bigram context
2 if the word is in the cache with the current bigram context.

The output is no longer a probability estimate, but the task of transforming the output to a proper probability estimate is left for the user. Obviously, information of the frequency of the cached words is lost this way. What we expect to benefit from this is that now we can explicitly make distinction between the three cases enumerated in the definition. The bin estimation method that is presented in chapter 3.5 is well suited for utilizing the cache information of this form. We will see that this formulation facilitates remarkable reduction in the number of parameters used in the bin estimation method. In chapter 4 both formulations are used in the experiments to compare their performance.

2.4 Latent semantic analysis

One proposed approach to utilize the knowledge about longer distance dependencies between the words is the latent semantic analysis (LSA). LSA is a paradigm that extracts salient relationships between words based on the information about which words tend to occur in the same context. (Bellegarda et al., 1996), (Bellegarda,

1997), (Coccaro and Jurafsky, 1998)

Mathematically the relevant part of the LSA is the singular value decomposition (SVD) that is used to reduce the dimensionality of the word-document co-occurrence matrix thus producing a compact representation of the relationships between words and documents. The use of SVD for information retrieval purposes was introduced in 90's as latent semantic indexing (LSI) by Deerwester et al. (1990).

LSA begins with a word-document co-occurrence matrix W where element $W_{i,j}$ is the number of times word w_i occurs in document d_j normalized by the document length. Not all words are semantically equally important. Many common words, like *and*, *or*, and *is*, can occur in any semantic context and do not carry much information about the content. Usually the matrix elements $W_{i,j}$ are weighted by the semantical importance of the word w_i. How to measure semantical importance is not trivial and several formulations have been suggested by the information retrieval community. Bellegarda (2000) uses normalized word entropy calculated by equation

$$\epsilon_i = -\frac{1}{\log N} \sum_{j=1}^{N} \frac{c_{i,j}}{t_i} \log \frac{c_{i,j}}{t_i}, \tag{2.23}$$

where $c_{i,j}$ is the number of times word w_i occurs in document d_j, $t_i = \sum_j c_{i,j}$ is the total number of times word w_i occurs in the whole data, and N is the number of the documents.

A simpler weighting function, which is also used in this work, is the document frequency defined as

$$\epsilon_i = \frac{\log q_i}{\log N}, \tag{2.24}$$

where q_i is the number of documents the word w_i occurs in and N is the total number of documents.

Using the weighting for words, the element $W_{i,j}$ is presented as

$$W_{i,j} = (1 - \epsilon_i)\frac{c_{i,j}}{n_j} \tag{2.25}$$

where n_j is the total number of words present in document d_j and ϵ_i is calculated using 2.23 or 2.24.

The i'th row of the matrix W can be interpreted as a feature vector of word w_i and correspondingly, the j'th column as a feature vector of document d_j. This feature presentation is, however, impractical since the dimension of the both vectors is very high, and most of the elements are zero. (Bellegarda, 2000)

Singular value decomposition (SVD) is a method from linear algebra that can be used to reduce the dimensionality of a matrix. SVD is closely related to eigenvector

decomposition and factor analysis. SVD gives a decomposition of $M \times N$-matrix W to three matrices U, S, and V as follows:

$$W \approx \hat{W} = USV^T \tag{2.26}$$

where
U $M \times R$ left singular matrix with row vectors u_i, $i \in [1, ..., M]$
S $R \times R$ diagonal matrix with singular values in descending order
V $N \times R$ right singular matrix with row vectors v_j, $j \in [1, ..., N]$
$R \leq min(M, N)$.
In the case $R = M = N$, $W = \hat{W}$. In our purpose, however, $R \ll min(M, N)$.

Both left and right singular matrices U and V are column-orthonormal, i.e., $U^TU = V^TV = I_R$. This means that the column vectors of U and V both define an orthonormal basis for the space of dimension R spanned by the row vectors of matrices U and V, correspondingly.

SVD has profitable properties that make it appealing in reducing the dimension of the co-occurrence matrix W. It can be shown that matrix $\hat{W}$ is the best rank-R approximation to the co-occurrence matrix W for any unitarily invariant norm. The column vectors of matrices U and V span the new feature space S of dimension R. In this feature space, word w_i is presented as u_iS and document d_j as v_jS, where u_i and v_j are the i'th and the j'th row vectors of matrices U and V, correspondingly. This means that the words and the documents are presented as linear combinations of latent features that arise from the SVD mechanism. The features are such that they minimally span the words in the vocabulary $\mathcal{V}$ and the documents in the corpus $\mathcal{T}$.

SVD provides us a vector presentation that depicts the semantic information of the words and the documents. The semantic relation between a word and a document can be measured by their distance in the space S using some suitable distance measure. The element $W_{i,j}$ of the co-occurrence matrix W depicts the extent to which word w_i and document d_j co-occur in the training data. Note that $W = USV^T = US^{1/2}S^{1/2}V^T$. So the cell (i, j) of the matrix W can be obtained as a dot product of the i'th row of $US^{1/2}$ and the j'th row of $VS^{1/2}$, written as $u_iS^{1/2}$ and $v_jS^{1/2}$. Giving this reasoning, Bellegarda suggests the dot product

$$K(w_i, d_j) = \cos(u_iS^{1/2}, d_jS^{1/2}) = \frac{u_iSd_j^T}{||u_iS^{1/2}||\,||d_jS^{1/2}||} \tag{2.27}$$

as a natural "closeness" measure between words and documents. Value $K(w_i, d_j) = 1$ means that word w_i is strongly related to the document d_j, while $K(w_i, d_j) << 1$ means that the word is not related to the topic of the document. Rigorously, 2.27 is not a proper distance measure as it gives also negative values and the distance of a vector to itself is not zero. A proper distance measure can be derived from it, for example, by taking arcus cosine of $K(w_i, d_j)$ defined in equation 2.27. In this work,

the values are simply shifted and scaled to range $[0, 1]$, and the distance measure is defined as

$$K(w_i, d_j) = \frac{1}{2}(1 + \frac{u_i S d_j^T}{||u_i S^{1/2}||||d_j S^{1/2}||}) \tag{2.28}$$

During testing, the history vector $\tilde{d}$ is constructed from the words that have been seen in the document so far. The history vector is the weighted average of the word feature vectors where the words are weighted using equation 2.23 or 2.24. The history vector is calculated using the update rule

$$\tilde{d_k} = \frac{1}{n_k}[(n_k - 1)d_{k-1} + (1 - \epsilon_i)u_i]. \tag{2.29}$$

presented by Bellegarda (2000). Bellegarda uses also exponential decay to progressively discount the older utterances. The modified formula is

$$\tilde{d_k} = \frac{1}{n_k}[\lambda(n_k - 1)d_{k-1} + (1 - \epsilon_i)u_i], \tag{2.30}$$

where $0 < \lambda \leq 1$. In this work the documents are considered homogeneous and λ is set to one.

Having defined the distance measure K and the history vector $\tilde{d}$, the last step is to transform the distance measure $K(w, \tilde{d})$ to a proper probability estimate $P(w|\tilde{d})$. Various formulations have been suggested. Coccaro and Jurafsky (1998) used formulation

$$P(w_i|\tilde{d}) = \frac{(\cos(w_i, \tilde{d}) - \min_j \cos(w_j, \tilde{d}))^\gamma}{\sum_j \cos(w_i, \tilde{d}) - \min_j \cos(w_i, \tilde{d})}, \tag{2.31}$$

where the exponent γ is used to expand the dynamic range of the distribution. Bellegarda (2000) used some kind of multivariate distribution estimation. In this work a simplified version of 2.31 is used:

$$P(w_i|\tilde{d}) = \frac{K(w_i, \tilde{d})^\gamma}{\sum_j K(w_j, \tilde{d})^\gamma} \tag{2.32}$$

where the exponent γ is optimized on the training data. The dynamic range of the dot product 2.27 is narrow what for the exponent γ is used to give better differentiation between words.

LSA is a versatile framework and its range of use is not limited to what has been presented here. SVD gives a compact feature presentation for words and documents. Bellegarda utilizes this aspect in clustering words and documents to semantically motivated clusters. Using both word and document clustering in smoothing the probability distributions, Bellegarda achieved 53 % perplexity reduction to the bigram baseline while without clustering the reduction was 32 %. With trigrams the perplexity reductions were 33 % and 19 %, correspondingly.

2.5 Topic model

Gildea and Hofmann (1999) proposed a topic model that presents topic information as latent variables. The word probability is given as

$$P(w|h) = \sum_{t=1}^{T} P(w|t)P(t|h). \tag{2.33}$$

Here t is a latent variable that is supposed to refer to different topics. $P(w|t)$ is the topic-specific probability distribution and $P(t|h)$ is the mixing weight that depends on the history. From 2.33 we see that the topic information affects only the unigram probabilities. However, this is not a principled limitation of the model.

Topic Decomposition

The topics are learned automatically from the data using an EM-algorithm and no hand-labeled topic information is used. However, the number of the topic variables must be decided. The data is seen as unigrams and the higher order structures are not considered by the model. So the data can be expressed as a word-document matrix W where the element $W(i, j)$ denotes how many times word w_i occurs in document d_j. The job is to find distributions $P(w|t)$ and $P(t|d)$ that maximize the log-probability of the training data

$$l(\Theta; W) = \sum_{<w,d>} W(w, d) \log \sum_{t} P(w|t)P(t|d). \tag{2.34}$$

Here Θ refers the to the model parameters $t \in \{1, \dots T\}$.

In the training phase, the parameters $P(w|t)$ and $P(t|d)$ are initialized randomly. The EM-algorithm involves two steps that are repeated. In the E-step, the posterior probabilities of the latent variables $P(t|w, d)$ are calculated using the model parameters $P(w|t)$ and $P(t|d)$. In the M-step, the parameters $P(t|d)$ are re-estimated.

In the E-step the probability that a particular word w in the document d was generated by the topic factor t is calculated. For the r'th iteration Bayes' rule yields

$$P^{(r)}(t|w, d) = \frac{P^{(r-1)}(w|t)P^{(r-1)}(t|d)}{\sum_{t'=1}^{T} P^{(r-1)}(w|t')P^{(r-1)}(t'|d)}. \tag{2.35}$$

In the M-step the model parameters are recalculated given the values for the latent variables $P^{(r)}(t|w, d)$ that were calculated in the previous E-step:

$$P^{(r)}(w|t) = \frac{\sum_d n(w, d)P^{(r)}(t|w, d)}{\sum_{w'} \sum_d n(w', d)P^{(r)}(t|w', d)}, \tag{2.36}$$

$$P^{(r)}(t|d) = \frac{\sum_w n(w,d)P^{(r)}(t|w,d)}{\sum_{t'}\sum_w n(w,d)P^{(r)}(t'|w,d)}. \tag{2.37}$$

In the original paper (Gildea and Hofmann, 1999), a modified version of E-step was used to prevent overfitting.

Using the model

In testing, the job is to estimate the mixing portions $P(t|h)$ as the current document is revealed. $P(t|h)$ can be determined by keeping the probabilities $P(w|t)$ fixed while estimating $P(t|h)$ and iterating 2.35 and 2.37 over the words seen in the current document so far. Doing the full EM calculation each time would be time consuming. Instead, an online approximation is used to calculate $P(t|h)$:

$$P(t|h_i) = \frac{1}{i+1}\frac{P(w_i|t)P(t|h_{i-1})}{\sum_{t'} P(w_i|t')P(t'|h_{i-1})} + \frac{i}{i+1}P(t|h_{i-1}), \tag{2.38}$$

$$P(t|h_1) = P(t) = \frac{\sum_{w,d} n(w,d)P(t|d)}{\sum_{w,d} n(w,d)}. \tag{2.39}$$

Gildea and Hofmann state that using full EM iterations gave only negligible improvement with higher computational cost. Once the topic mixing weights $P(t|h)$ have been determined, word probabilities can be calculated using the formula 2.33.

2.6 Morph based language models

The large number of words caused by inflection produces severe problems in language modeling. Any vocabulary of practical size cannot cover adequately all words, and the out-of-vocabulary-rate will be intolerably high. Also, the large vocabulary worsens the problem of data sparsity. The larger the vocabulary, the more data is needed to train the language model properly and the resulting language models will be larger. To overcome these problems, Creutz and Lagus (2002) presented two algorithms to automatically split words into smaller units "morphs" that approximate the natural morphemes. Using the morphs, the size of the vocabulary can be reduced dramatically. Now different inflectional forms of the same word need not to be included in the dictionary, since the words are presented as concatenations of the morphs. For example, the Finnish word *autonkuljettajallekin* is divided to morphemes as auto|n|kuljetta|ja|lle|kin. Siivola et al. (2003) have used morphs in automatic speech recognition and show that using morphs as the basic language model units yields superior performance compared to using words or syllables.

In this work, the morphs are learned from the training data by the algorithm presented by Creutz and Lagus (2002). The algorithm tries to construct a model that minimizes the description length of the data. This means that both the lexicon, i.e. the set of morphs, and the data, presented as the morphs, should be as compact as possible. For the detailed description of the algorithm, see (Creutz and Lagus, 2002).

The morphs are used as basic units in all language models in this work. Here the corpus was preprocessed so that the words that occurred only once were removed. Rest of the words were included to the corpus only once. So instead of minimizing the description length of the data, the description length of the vocabulary was minimized, since this yields smaller lexicon which tends to produce better results in speech recognition.

A problem that arises in speech recognition when using sub-word units instead of whole words is that we no longer know where the previous word ends and the next one begins. In most cases, continuous speech does not give any acoustical information about the word boundaries. So the word boundaries have to be determined by the language model. In the training data the word boundaries are marked with symbol $<w>$ and treated as normal morphs. During recognition, for each hypothesis also another hypothesis, where a word boundary is added to the end, is created.

2.7 Handling OOV words

Any vocabulary of practical size does not contain all words with different word forms. Thus it is possible that the test data contain words that are not in the vocabulary. The language model would give zero probability to such words making the perplexity go infinite. In practice, OOV words are skipped and the perplexity is calculated only over the words that exist in the vocabulary. Having a small vocabulary yields low perplexities as the probability mass is shared only to a small number of words, but the OOV rate grows high. Thus, the perplexity value alone does not determine the quality of the language model but traditionally the perplexity value and the portion of OOV words (OOV rate) in the test data are both given.

Sometimes the problem of OOV words is solved so that all OOV words are mapped to a single symbol "<unk>". A probability for the <unk> symbol is then estimated as for any other word. However, if the vocabulary is chosen to contain all the words in the training data, then the probability for the <unk> symbol can not be estimated this way. In that case, an arbitrary probability value is assigned to <unk> (Clarkson and Robinson, 1997).

Another solution for determining probability estimates for the unseen words is arises from the use of the sub-word units. Words that are not in the vocabulary are split into

smaller units that exist in the vocabulary. This way, to any word can be calculated a probability, regardless of whether it has been seen in the training data. This means that we have virtually an unlimited vocabulary. This approach is used also in this work.

2.8 Evaluating the language models

The quality of a language model is best evaluated by measuring the performance of the target system that it is used in. In speech recognition this means measuring the word or phoneme error rate in speech recognition experiments. However, running a lot of speech recognition tests is time consuming. Optimizing a possibly large set of language model parameters would also need a large amount of speech data which is not always available. Thus, we need faster ways to evaluate language models.

The most common measure for evaluating language models is perplexity (ppl) calculated over a text, defined as

$$\text{ppl} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1^{i-1})}} \tag{2.40}$$

where N is the number of the words in the data. Perplexity measures how well the model predicts the given word sequence. For example, perplexity of a model that gives the same probability for each word equals to the number of words in the vocabulary. So the perplexity can be figured as being approximately the number of equally probable words that the model has to choose from. The better the language model the lower the perplexity. Perplexity value 1 means that the model predicts the following word with certainty.

It is known that perplexity does not predict word error rate in speech recognition systems very well and several alternatives have been suggested. Bahl et al. (1989) claim that recognition errors are strongly correlated with low-probability language model predictions and suggest using the percentage of the text that was predicted with low probability. Chen et al. (1998a) show that perplexity is a good predictor of the word error rate for conventional n-gram models that have been trained with data from the same domain as the data that the tests are run with. They also present two measures that outperform the perplexity on other than n-gram models. *M-ref* tries to extend the perplexity by relaxing the assumption that the function between the perplexity and the word-error rate is linear. The other measure imitates the word error calculation by artificially generated speech recognition lattices. Clarkson and Robinson (1997) show that a combination of the log probability and the entropy leads to a measure that correlates with word-error rate better than the perplexity. For more information on different measures for language model quality, see Chen et al. (1998a).

In this work, perplexity is used to measure the language model quality. Despite of its drawbacks, perplexity is the most commonly used measure. So the results are more easily compared with other researchers' work. The language models and the parameters used in the combination methods are optimized to minimize the perplexity over the training data and the development data, correspondingly. All results are given as perplexities or relative perplexity reductions compared to the baseline.

Usually the language model perplexity is calculated over words as defined in equation (2.40). When using morphs or other sub-word units instead of whole words this has to be taken into account, and the measure is re-written as

$$\text{ppl} = \sqrt[N]{\prod_{i=1}^{L} \frac{1}{P(w_i|w_1^{i-1})}} \tag{2.41}$$

where N is the number of words as in equation (2.40) and L is the number of tokens, i.e. morphs + word boundaries. This way the perplexity is independent of the choice of the morph set. The absolute perplexity values are actually not very interesting since in addition to the language model, they depend also on the test data. In this work we are more interested in the relative perplexity reductions compared to a well-reasoned baseline.

Chapter 3

Combination methods

The major target of this thesis work is to study methods for combining different language models together. In this chapter, five methods that are used in the literature are described. In addition, a new approach based on histogram estimation of the likelihood density function is introduced.

The five discussed combination methods are backoff, linear interpolation, log-linear interpolation, unigram rescaling, and maximum entropy modeling. In the backoff method the models are sorted and the best model is used whenever possible. In the linear interpolation the probability is calculated as a weighted sum of the component probabilities. In the log-linear interpolation the component values are scaled with suitable exponent values and multiplied together, and in the unigram rescaling method the product is in addition divided by the unigram probability.

The backoff and the linear interpolation are more or less heuristic methods as it is not straightforward to give a probabilistic interpretation to them. Bellegarda (2000) shows that the unigram rescaling can be derived for combining the LSA and the n-gram model together under relatively mild assumptions. The log-linear interpolation can be seen as a special case of the unigram rescaling when uniform prior distribution is used. In the bin estimation method the likelihood for a word to occur given the component probabilities is estimated using histograms. All these methods are blind in the sense that they only take the output values of the models as inputs and do not utilize any context information. A completely different approach is taken in the maximum entropy modeling where a single model is created to incorporate the different information sources using the maximum entropy principle.

3.1 Backoff

Backoff is a simple but efficient combination method originally proposed by Katz (1987) for combining different order n-gram models together. In backoff method, The language models are ordered from the most specific to the most general model. For n-gram models the highest order n-gram model is naturally the most specific model and the unigram model is the most general model. The specific model uses more information than the general model and thus the specific model is used whenever possible. If the probability for an event is missing from the specific (higher order) model we back off to the more general (lower order) model. The backoff model is defined recursively as:

$$P(w_i|h) = \begin{cases} P_Q(w_i|h), & \text{if } P_Q(w_i|h) > 0 \\ B_h P_{Q-1}(w_i|h), & \text{if } P_Q(w_i|h) = 0 \end{cases} \tag{3.1}$$

where $P_Q(w_i|h)$ is the probability estimate of the order Q model and B_h is the backoff weight that is calculated for each context h to make the conditional probabilities to sum up to 1.

When using the backoff method, some discounting method must be used to shift some of the probability mass to the unseen events[1]. Otherwise there would be no probability mass left for the words that are not present in the higher order model and thus the lower order n-gram would be used only in the case that the context h was not present in the higher order model.

Backoff method has the advantage that it can be stored compactly and the probabilities can be calculated efficiently. Backoff method does not actually combine the information sources but rather chooses one among them. The lower order models make no contribution to the probability estimate if the event is present in the higher order model. As the higher order model suffers more from the data sparsity, it might be useful to use also the lower order model even though the probability in the higher order model was available. Another drawback in the backoff method is that it exhibits a discontinuity around the point where the backoff decision is made. Chen and Goodman (1998) have shown that n-gram models consistently work better when the combination is done using linear interpolation than when using backoff.

Ordering the models according to their generality is straightforward with normal n-gram models but when more complex models, e.g. cluster or skipping models, are used, the correct order may not be trivial and the best backoff path has to be defined experimentally. Kirchhoff et al. (2002) have used the backoff method with factorial models where the number of possible backoff paths is huge. To find the optimal backoff path they have used genetic algorithms. They also presented a generalization to the backoff scheme using multiple backoff paths where the backing off can be done

[1] see section 2.1

using several paths at the same time. As the linear interpolation tends to outperform the backoff method with standard n-gram models it would be interesting to know whether the linear interpolation could be used also with the factorial models and how would the results compare to the reported ones.

The backoff method is not used in the experiments in this work. The major reason for this is that the probability distributions of the long-range models change each time a word is added to the history. This means that the backoff weights would have to be re-calculated after each word. When combining the 4-gram with the topic or the LSA model, the most natural order for the models would be 4-gram, 3-gram, 2-gram, topic/LSA, (and 1-gram). This would mean that the topic or the LSA model would be used only in the case that the bigram context was not present in the n-gram model. The proportion of such cases is quite small which might reduce the effect of the model. For the cache model the best ordering is not as obvious. Using the backoff method with the cache model would mean that some sort of discounting would be needed to shift some probability mass from the cached words to the unseen words to prevent zero probabilities. All this intricacy makes the backoff method an unattractive choice with the cache model. The purpose of the cache is to boost the probabilities of the occurred words rather than to create a complete language model. So it is not likely to be useful alone but should always be used together with some other model. As the backoff method does not actually combine the models but chooses one of them, the backoff method is not likely to work well with the cache model.

3.2 Linear interpolation

A simple and widely used combination method is linear interpolation. The linear interpolation means taking a weighted sum of the components as

$$P(w_i) = \sum_{k=1}^{K} \lambda_k P_k(w_i), \tag{3.2}$$

where P_k's are the component probability estimates for the word w_i and λ_k's are the interpolation weights satisfying the constraints $0 <= \lambda_k <= 1$ and $\sum_{k=1}^{K} \lambda_k = 1$. The interpolation weights are optimized on the held-out data.

An advantage of the linear interpolation is that it is simple and fast to calculate. If the inputs are probability estimates, also the output is a probability estimate. Linear interpolation is also guaranteed not to give a worse estimate than the worst of its components. In practice, if the test data is similar enough to the data that the weights were optimized on, the linear interpolation performs at least as well as the best component model.

Linear interpolation can be interpreted as a weighted average of all models. Though the linear interpolation is simple, it is hard to give a probabilistic justification to it, but the method is rather heuristic way of combining the language models. In practice, the linear interpolation has been found to work well in many cases. For example, Goodman (2000) has used linear interpolation in combining several language models together obtaining remarkable perplexity reduction over the single models.

A generalization to the basic idea is to use variable interpolation weights (Jelinek and Mercer, 1980), (Rosenfeld, 1994). The weights could be adjusted on runtime based on, for example, the estimation how reliable each model is in the current context. Kalai et al. (1999) have used three methods for determining the interpolation weights on-line achieving notably better perplexity results than using static interpolation weights when using inhomogeneous data.

In this work, static interpolation weights are used. The linear interpolation method is used in combining each of the topic, LSA, and the cache model with the 4-gram model. The topic model tries to reveal the underlying word distribution that is affected by the current topic. Remember, that in the standard interpolated n-gram model the unigram probability is taken into account by interpolating it with the higher order n-grams. Thus it seems reasonable to assume that interpolating the topic dependent unigram probability of the topic model with the n-gram model should improve the result. On the other hand, the word distribution of the topic model is much more uniform than the distribution of the n-gram model. Interpolating the n-gram model with the topic model will cause distribution smoothing compared to the plain n-gram distribution thus weakening the discriminative capability of the model. So it is likely that the optimal interpolation weight of the topic model will be quite low which is also the contribution of the model in that case. Similar consideration applies also to the LSA model. The cache model is quite different from the topic and the LSA model. One interpretation of interpolation of the n-gram and the cache model is that a part of the probability mass is reserved for the cache model to boost the probabilities of the words that have been seen in the history.

3.3 Log-linear interpolation

Another simple combination method is log-linear interpolation, defined by equation

$$p(w_i) \sim \prod_{k=1}^{K} P_k(w_i)^{\lambda_k} \tag{3.3}$$

where λ_k's are scaling parameters that adjust the contribution of each component model. Instead of summing the probabilities as in linear interpolation, the probabilities are multiplied together. This corresponds to linear interpolation done in the

logarithmic scale, where comes the name for the method. If $\lambda_k = 0$ then $P_k(w_i)^{\lambda_k}$ is one for all words $w_i \in \mathcal{V}$, and so the k'th model does not affect the total probability at all.

The motivation behind the log-linear interpolation lies in the better interaction of the components compared to the linear interpolation. When both components are large, the result is larger than either of its components. Respectively, when both components are small, the result is smaller than its components. This can be justified by an example. Let us assume that we have a trigram model that gives a relatively large value for a certain word. We have also a topic-based unigram model that gives the word a raised probability compared to its global frequency, but lower than the probability determined by the trigram model. Now it seems reasonable to assume that the true probability of the word is higher than the probability given by the trigram model.

A special case of the log-linear interpolation is the geometric mean, when $\sum_k \lambda_k = 1$ and $0 <= \lambda_k <= 1$. However, we are not obliged to set such constraints to λ's.

A drawback in the log-linear interpolation is that the output is no longer a probability estimate as the sum of the values over all words is not necessarily one. Thus the result must be normalized by equation

$$P(w_i) = \frac{p(w_i)}{\sum_{i=1}^{N} p(w_i)} \tag{3.4}$$

Normalization requires calculating the probabilities for all words in the vocabulary and is thus computationally expensive. However, in speech recognition system the constraint that a pure probability model is needed may possibly be loosened. What we need is a number that describes the relative likelihood of a word so that it is comparable with other words also in different contexts. The effect of omitting the normalization in speech recognition is discussed in chapter 4.5.

3.4 Unigram rescaling

The unigram rescaling method is expressed by formula

$$p(w_i|h) \sim \frac{P_1(w_i|h)P_2(w_i|h)}{P(w_i)} \tag{3.5}$$

where P_1 and P_2 are the component model probabilities and $P(w_i)$ is the normal unigram probability. Gildea and Hofmann (1999) presented the method for integrating the topic model with the n-gram model and showed that it outperformed the linear and the log-linear interpolation. Bellegarda (2000) used the method in combining the

LSA model with the n-gram model and showed that the formula 3.5 can be derived for the LSA model and the n-gram model under relatively mild assumptions. The derivation is repeated here. The derivation starts from the overall language model probability

$$P(w_i|h_{i-1}^{N+L}) = P(w_i|h_{i-1}^{N}, h_{i-1}^{L}) \tag{3.6}$$

where h_{i-1}^{N} means the n-gram context w_{i-n+1}^{i-1},h_{i-1}^{L} means the history seen by the LSA-model, and h_{i-1}^{N+L} means the integration of the two. The expression can be rewritten as

$$P(w_i|h_{i-1}^{N+L}) = \frac{P(w_i, h_{i-1}^{L}|h_{i-1}^{N})}{\sum_{w_j \in \mathcal{V}} P(w_j, h_{i-1}^{L}|h_{i-1}^{N})} \tag{3.7}$$

where $\mathcal{V}$ refers to the vocabulary. The numerator in equation 3.7 can be expanded and rearranged to

$$P(w_i, h_{i-1}^{L}|h_{i-1}^{N}) = P(w_i|h_{i-1}^{N})P(h_{i-1}^{L}|w_i, h_{i-1}^{N}) = P(w_i|w_{i-n+1}^{i-1})P(\tilde{d}_{i-1}|w_{i-n+1}^{i-1}) \tag{3.8}$$

where $\tilde{d}_{i-1}$ refers to the document constructed based on the revealed history h_{i-1}^{L} as is explained in sections 2.4 and 2.5. Now the assumption is made that the probability of the document history given the current word is not affected by the immediate context preceding it.

$$P(\tilde{d}_{i-1}|w_{i-n+1}^{i-1}) \approx P(\tilde{d}_{i-1}|w_i) \tag{3.9}$$

According to Bellegarda, if the document history is long enough, the semantic anchoring should be sufficiently strong and the assumption should hold. Thus the integrated probability becomes

$$P(w_i|h_{i-1}^{N+L}) = \frac{P(w_i|h_{i-1}^{N})P(\tilde{d}_{i-1}|w_i)}{\sum_{w_j \in \mathcal{V}} P(w_j|w_{i-n+1}^{i-1})P(\tilde{d}_{i-1}|w_j)}. \tag{3.10}$$

Using Bayes' rule

$$P(\tilde{d}_{i-1}|w_i) = \frac{P(w_i|\tilde{d}_{i-1})P(\tilde{d}_{i-1})}{P(w_i)} \tag{3.11}$$

and reducing $P(\tilde{d}_{i-1})$ from the equation we get

$$P(w_i|h_{i-1}^{N+L}) = \frac{P(w_i|w_{i-n+1}^{i-1})\frac{P(w_i|\tilde{d}_{i-1})}{P(w_i)}}{\sum_{w_j \in \mathcal{V}} P(w_j|w_{i-n+1}^{i-1})\frac{P(w_j|\tilde{d}_{i-1})}{P(w_j)}} \tag{3.12}$$

where $P(w_i)$ is the normal unigram probability of the word w_i. (Bellegarda, 2000)

The term $P(\tilde{d}_{i-1}|w_i)$ in 3.10 can be viewed as a prior probability on the current document history. So 3.10 translates the classical Bayesian estimation of the n-gram probability using a prior distribution obtained from LSA. (Bellegarda, 2000)

In practice, the expression 3.12 is often modified so that scaling factors are used to adjust the contribution of each model (Bellegarda, 2000)

$$P(w_i|h_{i-1}^{N+L}) = \frac{P(w_i|w_{i-n+1}^{i-1})^{\lambda_N} \frac{P(w_i|\tilde{d}_{i-1})^{\lambda_T}}{P(w_i)^{\lambda_{uni}}}}{\sum_{w_j \in} P(w_j|w_{i-n+1}^{i-1})^{\lambda_N} \frac{P(w_j|\tilde{d}_{i-1})^{\lambda_T}}{P(w_j)^{\lambda_{uni}}}} \tag{3.13}$$

where λ's are optimized on the development data.

Equation 3.12 was derived for the integration of n-gram and LSA models. The derivation is applicable with similar assumptions also for the topic model 2.5. The unigram rescaling method can be given an intuitive interpretation. The n-gram probability is modified by the topic model factor

$$\frac{P(w_i|\tilde{d}_{i-1})}{P(w_i)} \tag{3.14}$$

The topic model factor 3.14 is greater than one, if the word has a raised probability given the history, and less than one if the word has a lowered probability, compared to the global probability $P(w_i)$.

3.5 Bin estimation method

In this section, a new approach based on multivariate likelihood density estimation is introduced for combining language models. The derivation is given here for the case of two models. To generalize the approach to any number of models is straightforward.

The idea in the method is to determine the probability for a word given the estimates from the component models. To do this, the input space spanned by the component model outputs is divided to bins, and the likelihood value is estimated for each bin. Inside the bin the likelihood is assumed constant. The more dense grid the better this assumption holds. In the following, the formula for the likelihood is derived and an estimator for estimating the likelihood from data is presented.

Let us assume that we have two models M_A and M_B that produce probability estimates $f_A(w_i|h)$ and $f_B(w_i|h)$ for any word $w_i \in \mathcal{V}$. Now we want to know what is the likelihood for word w_i to occur in context h when the models give estimates $f_A(w_i|h) = a$ and $f_B(w_i|h) = b$. If the distributions are assumed to be continuous we have to consider short intervals $a \leq f_A < a+\epsilon$ and $b \leq f_B < b+\epsilon$ to avoid zero probabilities. This means that we divide the two dimensional probability space $f_A \times f_B$ to rectangular bins that are denoted as $B_r = \{(x_1, x_2) \,|\, a_r^1 \leq x_1 < a_r^2, b_r^1 \leq x_2 < b_r^2\}$. So what we are interested in, is the likelihood

$$p(w = w_i \,|\, f_A(w_i|h), f_B(w_i|h)) \approx p(w = w_i \,|\, (f_A(w_i|h), f_B(w_i|h)) \in B_r) =: p_{B_r}. \tag{3.15}$$

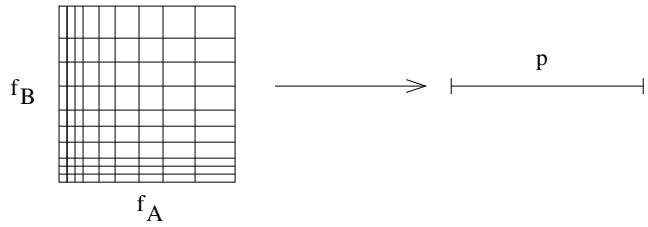

Figure 3.1: Mapping from probability space to likelihood

where bin B_r is the bin for which $(f_A(w_i|h), f_B(w_i|h)) \in B_r$.

The task is to estimate the likelihood defined by equation 3.15 for each bin from the data. The training data is simply a sequence of words, which can also be presented in form

$$(w_1, h_1), (w_2, h_2) \dots (w_L, h_L) \tag{3.16}$$

where h_k refers to the sequence of all preceding words $h_k = w_1 w_2 ... w_{k-1}$. To be able to estimate (3.15) we need to calculate $f_A(w_j|h_k)$ and $f_B(w_j|h_k)$ for all words w_j in the vocabulary in addition to the 'correct' one that actually occurs in the data. Let us use notation

$$x_k^j := (f_A(w_j|h_k), f_B(w_j|h_k)) \tag{3.17}$$

to refer to the values from models M_A and M_B for word w_j given the context h_k. Note that here the index j determines the word whereas the index k refers to the position in the training data. Further, we define sequence

$$f_k \in [1, 2, ..., N], k \in [1, 2..., L] \tag{3.18}$$

to refer to the index of the k'th word in the training data. Using this notation, the training data can be presented as sequence $(w_{f_1}, h_1), (w_{f_2}, h_2), ...(w_{f_L}, h_L)$.

To estimate the likelihood defined in (3.15) we use estimator

$$\hat{p}_L = \frac{\sum_{k=1}^{L} I(x_k^{f_k} \in B_r)}{\sum_{k=1}^{L} \sum_{j=1}^{N} I(x_k^j \in B_r)} \tag{3.19}$$

The likelihood is estimated independently for each bin. Next it will be shown that under some assumptions the estimator converges to the likelihood (3.15) that we want to estimate. The estimator, defined in equation 3.19, can be rewritten as

$$\begin{aligned} \hat{p}_L = \frac{\sum_{k=1}^{L} I(x_k^{f_k} \in B_r)}{\sum_{k=1}^{L} \sum_{j=1}^{N} I(x_k^j \in B_r)} = \\ \frac{N_L(w = w_i, (f_A(w_j|h_k), f_B(w_j|h_k)) \in B_r)}{N_L((f_A(w_j|h_k), f_B(w_j|h_k)) \in B_r)} \end{aligned} \tag{3.20}$$

Let us assume that the elements in the sequence 3.17 are independent. Let us further assume that there is a constant p'_{B_r} so that when $L \to \infty$ then 3.20 converges to it. Then, according to the frequentistic interpretation of probability,

$$\lim_{L\to\infty} \frac{N_L(w = w_i, (f_A(w_j|h_k), f_B(w_j|h_k)) \in B_r)}{N_L((f_A(w_j|h_k), f_B(w_j|h_k)) \in B_r)} = p'_{B_r} = \quad (3.21)$$

$$p_{B_r} = p(w = w_i \,|\, (f_A(w_i|h), f_B(w_i|h)) \in B_r) \quad (3.22)$$

and so we have shown that the estimator converges to the likelihood. The independence assumption of the elements in sequence 3.17 does not strictly hold. If we know the probability estimate $f_A(w_j|h_k)$ then we know that $f_A(w_t|h_k) \leq 1 - f_A(w_j|h_k)$ for all words w_t, $t \neq j$. However, when the size of the vocabulary is large, the independence should be a good approximation.

Equation (3.15) determines the likelihood for word w_i to occur given the model values. To transform this to a probability estimate the likelihood value has to be normalised so that probabilities for all words in a given context sum up to one. The normalisation is calculated by equation

$$P(w = w_i|(f_A(w_i|h), f_B(w_i|h))) = \frac{p_{B_i}}{\sum_{j:w_j\in\mathcal{V}} p_{B_j}}, \quad (3.23)$$

where B_i is the bin for which $(f_A(w_i|h), f_B(w_i|h)) \in B_i$, and B_j is the bin for which $(f_A(w_j|h), f_B(w_j|h)) \in B_j$.

Now we have derived the likelihood for word w_i to occur given the context and the models. This is done by splitting the space defined by the model values (f_A, f_B) to rectangular segments, call them bins, and doing the approximation that the likelihood is constant in each bin. We have defined an estimator to estimate the likelihood from data, and we have showed that under some assumptions the estimator converges to the likelihood we want to estimate.

It is worth to note that nothing is assumed about the relationship of the models M_A and M_B. Particularly, they are not assumed to be independent. Further, the models M_A and M_B need not to be probability models and their outputs do not have to belong between one and zero but any range of values is possible. Actually, in this case the assumption that x_k^j's, $j \in [1, ..., N]$ are independent is better justified. The property, that the model values need not to be probability estimates, may be useful in cases where it is uncertain how to transform the output of a model to a probability estimate. This property is utilized with the three-value cache model and also with the LSA model whose output is not a proper probability estimate since the normalization is omitted to save computation time.

In the training phase, the likelihood (3.15) is estimated for each bin. In the test phase, the model values $f_A(w_i|h)$ and $f_B(w_i|h)$ are calculated and the corresponding bin B_r for which $(f_A(w_i|h), f_B(w_i|h)) \in B_r$ is chosen to determine the likelihood.

Likelihoods for all other words are determined similarly, and the probability estimate is achieved doing the normalization by equation (3.23). In speech recognition, the normalization step is omitted to save computation time. The effect of this approximation is discussed in section 4.5.

Grid selection

The splitting of the input space into bins and using a constant likelihood value inside a bin introduces some quantization error. Tightening the grid would decrease the error but at the same time it would make the likelihood estimates in each bin more inaccurate as less data is left for each bin. Finding the optimal division means minimizing the error produced by these two error sources. Generally, there is no reason to be restricted on rectangular bins, and for example Gaussian mixtures could be used. In this work, however, a rather simple way for determining the grid is used. For more information on determining the division the reader is referred to see literature on estimation theory, e.g. (Silverman, 1986).

In this work, the division was determined independently for each dimension. The division was done as follows.

1. Probability values for all words in the development set are collected.
2. The division is chosen so that equivalent amount of the collected samples is assigned to each block. This may not be strictly possible as the probability estimators are not continuous in practice. This does not guarantee that equal amount of samples is assigned to each bin in the resulting multidimensional grid since the models are most likely not independent.
3. The block corresponding to the lowest probabilities is divided to four blocks. Most of the words w_k^j, for which $j \neq f_k$, get low probability estimates. So the total number of samples that fall into the low probability blocks is large and more dense grid can be used for the low probabilities. However, this depends on the model.

For all models, splitting the range $[0, 1]$ to 53 values was found to work well. The only exception is the three-value-cache. The three-value-cache has only three possible output values, which allows more dense division to be done for the 4-gram model. When combining the three-value-cache and the 4-gram model, the axis corresponding to the 4-gram values was divided to 250 parts.

A question that also arises from limited data is what to do with bins that have no data at all. Some positive number must be assigned to such bins, too. Some

heuristics can be used to estimate the value for such bins. For example, the value can be interpolated from the neighboring bins. This should not be a big matter since the model value pairs falling into such bins are rare concluding from their infrequency in the development data.

The major drawback of the method is that it needs quite a large amount of data to properly estimate the likelihood function, especially with several models and a dense grid. As the method has not been used before, we do not know how sensitive the method is for the choice of the grid.

3.6 Maximum entropy

All methods described above are blind methods in the sense that they do not use any information about the models that are to be combined. The outputs of each model are calculated independently and then the values are put into a formula to yield the combined probability estimate. A totally different approach arises from information theory. Maximum entropy (ME) method is based on the approach first proposed by Jaines (1957).

In ME approach, a single model is built so that it incorporates the information of the component models and assumes nothing more. In ME approach, language models, generally information sources, set constraints that the model has to satisfy. From all models that satisfy the constraints, the one which has the largest entropy is chosen.

How the constraints are presented in the maximum entropy approach is easiest to understand by an example. A bigram model says that the probability for a word given the previous word is $P(w_i|w_{i-1}) = a_{w_i,w_{i-1}}$. In maximum entropy approach, we loosen the requirement that the constraint $P(w_i|w_{i-1}) = a_{w_i,w_{i-1}}$ has to hold in every case. We accept that there may be also other things that affect the probability than just the previous word. Instead, we require that the constraint has to hold on average

$$\mathop{\mathbf{E}}_{h \textit{ ends in } w_{i-1}} [P(w_i|w_{i-1})] = a_{w_i,w_{i-1}}. \tag{3.24}$$

In language modeling, we are usually interested in conditional probabilities $P(w|h)$, like above. In this section, the main ideas of the ME modeling are represented for joint probabilities $p(w, h)$, and in the end it is shown how to calculate the conditional probabilities.

Any subset A_k of the event space S can be specified by an *index function* or *feature*

function $g_k(s)$:

$$g_k(s) = \begin{cases} 1 & \text{if } s \in A_k \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

Using the feature function, the constraint

$$\underset{s \in A_k}{E}[P(s)] = a_k \tag{3.26}$$

can be written as

$$\sum_{s \in S} p(s) g_k(s) = a_k. \tag{3.27}$$

In fact, the function g_k need not to be a binary function but any real-valued function defined on S will do. This means that any kind of information source can be brought into the ME model.

Constraints from any information source are given in the form of equation 3.27. The constraints define a *linear family* of probability distributions

$$\mathcal{P} = \{p : \sum_{s \in S} p(s) g_k(s) = a_k,\ 1 \leq k \leq K\} \tag{3.28}$$

Now our job is to find the distribution that maximizes the entropy

$$H(p) = -\sum_{s \in S} p(s) \log p(s) \tag{3.29}$$

from the linear family $\mathcal{P}$ defined in equation 3.28. Maximizing the entropy is equivalent to minimizing the *I-divergence*

$$D(p||q) = \sum_{s \in S} p(s) \log \frac{p(s)}{q(s)} \tag{3.30}$$

between p and the uniform distribution $q(s) = \frac{1}{|S|}, \forall s$. In this sense, the maximum entropy distribution is the smoothest distribution satisfying the given constraints.

Jaines (1957) and Csiszàr (1975) have shown that if the constraints are consistent, a unique ME solution is guaranteed to exist, and to be of the form:

$$p(s) = \prod_{k=1}^{K} \alpha_k^{g_k(s)} \tag{3.31}$$

The maximum entropy distribution can be found using the *generalized iterative scaling* (GIS) algorithm presented by Darroch and Ratcliff (1972).

Generalized iterative scaling

Generalized iterative scaling (GIS) algorithm seeks iteratively the probability distribution of exponential form

$$p(s) = \prod_{k=1}^{K} \alpha_k^{g_k(s)}, \tag{3.32}$$

which satisfies linear constraints of the form 3.27. The i'th iteration of the algorithm goes as follows:

1. Compute the expectations $\tilde{a_k}$ of the g_k's under the current probability distribution $p^{(i)}$:
$$\tilde{a}_k^{(i)} = \sum_{s \in S} p^{(i)}(s) g_k(s) \tag{3.33}$$

2. Update the parameters α_k
$$\alpha_k^{(i+1)} = \alpha_k^{(i)} \frac{a_k}{\tilde{a}_k^{(i)}} \tag{3.34}$$

3. Define the new probability distribution based on the new α's:
$$p^{(i+1)}(s) = \prod_k \alpha_k^{(i+1) g_k(s)} \tag{3.35}$$

If there is a solution to the set of constraints, the GIS algorithm is guaranteed to converge to the maximum entropy solution (Darroch and Ratcliff, 1972). GIS has, however, some practical problems. First, the convergence is quite slow for large models. Furthermore, intermediate values of model parameters may overflow or underflow the range of double precision before convergence. Della Pietra et al. (1997) have proposed the improved iterative scaling (IIS) algorithm to overcome these computational problems. However, ME is related to the maximum likelihood and more efficient optimization algorithms can be used in finding the model parameters. This is discussed more later in this chapter after the calculation of the conditional probabilities has been explained.

Conditional probabilities

In language modeling, we are usually interested in conditional probabilities $p(w|h)$ rather than joint probabilities $p(h, w)$. The conditional probability is written as

$$p(y|x) = \frac{1}{z(h)} \prod_{k=1}^{K} \alpha^{g_k(h,w)}, \quad z(h) = \sum_{w \in W} \prod_{k=1}^{K} \alpha^{g_k(h,w)} \tag{3.36}$$

To train the model parameter α's by GIS, in each iteration we need to compute the expectations of all features g_k under the probability distribution $p^{(i)}$, as

$$p[g_k] = \sum_{<h,w>} p(h,w)g_k(h,w) = \sum_{<h,w>} g_k(h,w)p(h)p(w|h), \text{ for } k = 1,\dots K, \quad (3.37)$$

For each feature g_k, we must find all <h,w>'s for which g_k is active. If any of the feature functions is real-valued and depends on the whole history, whose length is not bounded, then the number of possible histories is infinite. Even if the length of the history was bounded, the number of possible histories could be too large for enumerating them all.

A simple trigram model has $|\mathcal{V}|^2$ different histories, where $|\mathcal{V}|$ is the size of the vocabulary. Features corresponding to the unigram probability constraints are active for all histories and thus have $|\mathcal{V}|^2$ terms in the summation. This means that only for unigram features the number of calculations is $\sim |\mathcal{V}|^3$ for each iteration which is quite intractable. (Wu, 2002)

Berger et al. (1996) suggest using empirical distribution $\tilde{p}(h)$ obtained from training data as the marginal distribution in the calculation of the expectations 3.37 instead of the marginal distribution $p(h)$. Equation 3.37 is thus approximated by

$$p[g_k] = \sum_{h,w} p(h,w)g_k(h,w) \approx \sum_{h,w} \tilde{p}(h)p(w|h), \text{ for } k = 1,\dots,K \quad (3.38)$$

Now the training time is always limited by the length of the training data and the computational load reduces to a fraction, since most of the trigram contexts, for example, are never seen in the training data.

Feature selection for n-grams and cache

Next it will be described how to construct a ME model that incorporates information from n-gram models and a cache model. Each n-gram probability estimate $P(w_i|w_{i-n+1}^{i-1}) = a_{w_{i-n+1}^{i}}$ induces to the model a constraint

$$E[P(w_i|w_{i-n+1}^{i-1})] = a_{w_{i-n+1}^{i}}. \quad (3.39)$$

where $a_{w_{i-n+1}^{i}}$ is, for example, the maximum likelihood estimate of the n-gram probability $P(w_i|w_{i-n+1}^{i-1})$ in the training data. To write the constraint 3.39 in form 3.27 we have to define a feature function g_i so that $g_i(h,w)$ is 1 when $w = w_i$ and history h ends in word sequence w_{i-n+1}^{i-1}, and 0 otherwise. For unigram probability $P(w_i)$ the constraint is presented as

$$g_{w_i}(w,h) := \begin{cases} 1 & \text{if } w = w_i, \\ 0 & \text{otherwise} \end{cases} \quad (3.40)$$

Using 3.40, the constraint 3.39 can be written as

$$E_{<h,w>}[P(w_i|h)] = E_{<h,w>}[g_{w_i}(h,w)P(w|h)] = \sum_{<h,w>} g_{w_i}(h,w)P(w|h)P(h) = a_{w_i}. \tag{3.41}$$

For a bigram constraint, the feature function is defined as

$$g_{\{w_i,w_j\}}(h,w) := \begin{cases} 1 & \text{if } w = w_i \text{ and } h \text{ ends in } w_j \\ 0 & \text{otherwise} \end{cases} \tag{3.42}$$

For trigrams, the feature function is

$$g_{\{w_i,w_j,w_k\}}(h,w) := \begin{cases} 1 & \text{if } w = w_i \text{ and } h \text{ ends in sequence } w_j w_k \\ 0 & \text{otherwise} \end{cases} \tag{3.43}$$

Similarly, the constraints corresponding to n-grams of arbitrary order can be represented following the notation above.

Integrating a bigram cache model into the ME model requires two sets of constraints. The unigram cache constraints can be presented as

$$g_{cache\_w_i}(h,w) := \begin{cases} 1 & \text{if } w = w_i \text{ and } w \in h \\ 0 & \text{otherwise} \end{cases} \tag{3.44}$$

and the bigram cache constraints are presented as

$$g_{cache\_\{w_i,w_j\}}(h,w) := \begin{cases} 1 & \text{if } w = w_i, \;\; h \text{ ends in } w_j, \text{ and } \{w_j w_i\} \in h \\ 0 & \text{otherwise} \end{cases} \tag{3.45}$$

This implementation does not take into account how many times the word has occured in the history. So the presented formulation corresponds to the three-value cache model described in section 2.3.

Relation to maximum likelihood

The problem of finding the model $p^* \in \mathcal{P}$ that maximises the entropy has a dual problem of finding the model $q_\Theta(w|h)$ that maximises the likelihood of the training data (Berger et al., 1996). So the problem can be viewed as a traditional unconstrained optimisation problem where we maximise the log likelihood

$$L(\Theta) = \sum_{w,h} q_\Theta(h,w) \log q_\Theta(w|h) \tag{3.46}$$

where

$$q_\Theta(w|h) = \frac{\prod_{k=1}^{K} \alpha_k^{g_k(h,w)}}{\sum_{w \in \mathcal{V}} \prod_{k=1}^{K} \alpha_k^{g_k(h,w)}} \tag{3.47}$$

and the summation is calculated over the training data. As explained before, an empirical distribution can be used for the marginal probability $q_\Theta(h, w)$. Now any optimization method such as gradient methods can be used to solve the problem. Malouf (2002) has compared different algorithms for estimating the parameters for the conditional ME models and observed that conjugate gradient methods and variable metric methods are remarkably faster than GIS or IIS.

Even though the ME model produces the smoothest possible distribution that satisfies the given constraints, it is still prone to overfitting. The ME model constructed from the n-gram constraints has the same number of parameters as the corresponding n-gram model. Also, if no discounting is done to the feature expectations, the ME model is going to learn over-estimated probabilities just like the traditional n-gram models. Chen and Rosenfeld (2000) have compared different smoothing methods for ME models and report that a method based on Gaussian prior performed well in the task. They found that the ME model constructed using the Gaussian prior outperformed the traditional Kneser-Ney smoothed n-gram model.

Chapter 4

Experiments

Two sets of experiments are run in this work. First, the language model combinations are evaluated by calculating the perplexity over the test data. Second, the combinations are used in the speech recognition experiments. In both experiments the data is Finnish news articles. However, the speech data is from different time period than the text data that is used in training the language models and optimizing the combination methods. For this reason the language model perplexities are also calculated for the transcription of the data used in the speech recognition experiments.

All long-scale models, i.e. topic model, LSA model, and cache model, are combined with the 4-gram model one at time. The evaluated combination methods are linear interpolation, log-linear interpolation, unigram rescaling, and the bin estimation method. All combination methods are applied to combine each long-scale model with the 4-gram model. The only exception is the cache model. The cache model gives a zero probability for all words that have not occurred in the observed history. This is not a problem when the cache model is combined with the n-gram model by linear interpolation since the resulting probability is always greater than zero. When the cache model is combined using log-linear interpolation or unigram rescaling, the resulting probability will be zero for all words that have not been seen in the history. This could be avoided, for example, by giving a nonzero probability for the words that are not in the cache. In this work, this is not done and the log-linear interpolation and the unigram rescaling are not used with the cache model.

In addition to the two model combinations, some experiments are also run with three and four model combinations. When the number of models is raised, the number of possible combinations becomes large. Many of the combinations can be discarded based on the results of the two model combinations. Only the most appealing combinations are evaluated of which the best performing ones are presented here.

The bin estimation method is easily generalized to the case of more than two models. However, the number of bins, i.e. free parameters, grows rapidly when more models are brought in. Here the bin estimation method is used to combine the 4-gram model, the topic model, and the three-value-cache model together. The three-value-cache is used instead of the regular cache model to keep the number of the bins tolerable. The combination has still three times more bins than the two model combinations. To compensate this, three times more data is used to train the combination parameters in this particular case. So this experiment is not strictly comparable to the other experiments. However, the other studied combination methods might not benefit from increasing the amount of training data since they have only few parameters that will probably achieve the optimal values or close enough with less data.

Speech recognition tests are run with the same model combinations as the perplexity experiments. The speech data is stylistically very similar to the data used in training the language models. However, the news articles of the speech data are from different years than the training data. This may cause some performance reduction as the long range models are presumably effective in predicting the content words. To see how well the language models and the combination methods work with the speech data, perplexity experiments are run also for the transcription of the speech data.

4.1 Speech recognition system

The speech recognition system used in this work has been developed in the laboratory of computer and information science in HUT. A short description of the system is given below. For the detailed description of the system, see (Hirsimäki, 2002) and (Pylkkönen, 2004).

The speech recognition system consists of feature extraction, acoustic model, lexical model, language model, and decoder. Feature extraction transforms speech signal to a number sequence that contains relevant information respect to speech recognition. Speech signal is sampled at 16 kHz frequency and 16 mel frequency cepstral coefficients (MFCC), 16 delta coefficients and the power are extracted at each time.

The prevailing acoustic modeling paradigm in speech recognition has been for many decades the hidden Markov model (HMM). It is a simple model that allows the search of the best matching path to be done efficiently. In the HMM framework, acoustic models are created for each phoneme. For a phoneme, several models can be created for different acoustic contexts. Typically, triphones are used in contemporary speech recognition systems. This means that different models are used depending on the phoneme right before and right after the current one. Because of coarticulation, also longer phoneme contexts can have significant effect on the pronunciation of the phoneme and separate models could be created depending on the longer phonetic

context. However, when longer context is taken into account the number of models grows enormously, and huge amounts of data would be needed to train such models. For this reason, triphones are a common choice, and even then, some clustering has to be done to reduce the number of models.

Each model consists of a number of HMM states. Typically three states are used to model the beginning, the middle, and the end part of the phoneme. Each HMM state emits a feature vector with a certain probability. These probability distributions are usually modeled by Gaussian mixtures. In the training phase, parameters and weights for the Gaussian densities, and probabilities for the state transitions inside a model are optimized. In recognition, the word sequences that acoustically match best to the speech signal are found using the Viterby search.

Lexical model defines the vocabulary of lexical units, i.e. words, word sequences, or sub-word units, that the recognizer is able to recognize. Further, it describes how the words are constructed from the phonemes. In Finnish, building the lexicon is straightforward as the letters are uniquely mapped to phonemes with only few exceptions.

In the center of the speech recognition system is the decoder. The decoder takes care of the Viterby search and the language model computation, and combines these two. Architectures used in the speech recognition systems differ, for example, in how the language model is incorporated. In some architectures, the language model can be used already during the Viterby search. Such a solution, however, restricts the form of the language model. Our decoder is architecturally a *stack decoder*. The decoder stores the best hypothesises in stacks, so that for each time frame there is a separate stack. The hypothesises in a stack are expanded by words that have found to best match the speech by Viterby search. A new hypothesis is inserted to the stack that belongs to the most probable ending frame for the word. (Hirsimäki, 2002)

An advantage in the stack decoder is that it suites well for long-range language models. The recognized word history is maintained in a tree structure, so that each hypothesis has a uniquely defined word history. This allows us to easily integrate the examined long-range language models into the system. The state of the language model, i.e. the representation of the recognized word history, is stored together with the hypothesis. When the hypothesis is expanded with a new word, the state is updated and stored with the new hypothesis. The only modification made to the decoder in this work was the handling of the language model state.

4.2 Data

A text corpus of Finnish news articles from STT (Finnish National News Agency) is used for training and evaluating the language models and the combination methods. The text corpus comprises about 16.4 million words in 91 000 articles that are divided to 8 categories. The average document length is 180 words.

The text corpus is divided to three parts. 14 million words in 88674 documents are used for training the models (training set), 165 000 words in 1029 documents are used for optimizing the combination method parameters (development set) and 200 000 words in 1312 documents are reserved for the perplexity tests (test set). The rest of the data was left out for future purposes.

Speech recognition experiments are run with read STT news articles from years 1988-1992. The speech data consist of 288 articles of about one minute length. 3.7 hours is used in training the acoustic models, 30 minutes is used in tuning the combination method parameters and the language model scaling factor that adjusts the balance between the language model and the acoustic model, and 40 minutes is used for the tests. The speech data is provided by Inger Ekman and the department of Information studies at the University of Tampere.

Usually the data used for optimizing the combination parameters is some held out data that has not been used in training the models. A question that arises is whether the same training data could be used for training both the language models and the combination methods. With methods that have few parameters this is not an issue since the optimal parameters can be reliably determined with a small amount of data. The bin estimation method may need significantly more training data and thus the possibility to use the same data for both cases might be valuable. However, this is not studied in this work, and separate data sets are used for the training.

4.3 Training of the combination methods

For the perplexity experiments, all the interpolation and scaling weights in linear interpolation, log-linear interpolation, and unigram rescaling methods were optimized on the development data using Powell algorithm. The bin estimation method differs from the others in that it does not have interpolation or scaling weights but a number of bins for which the likelihood value is estimated.

In this work, 253x3 grid was used for the combination of the 4-gram model and the three-value-cache model. For all other two model combinations 53x53 grid was used. The grid was formed for each combination pair separately using the algorithm

presented in section 3.5. The likelihood values for each bin were estimated from the development data, and gentle filtering in two dimensions was applied for smoothing the values.

When combining the LSA model and the 4-gram model the unnormalized LSA outcomes were used as inputs for the bin-method. In the case of the topic model, the outcomes that were divided by the unigram probability of the particular word were used. This choice was made based on the preliminary experiments that showed that for the topic model the scaling by the unigram probability had remarkable effect to the results while for the LSA model this was not so important. When combining the cache model and the 4-gram model together using the bin-method, two different cache formulations were used as was described in section 2.3.

The preliminary experiments on the development set of the speech data showed that the parameters optimized on the text data did not work in speech recognition. So the combination method parameters were adjusted based on the development set of the speech data. Thorough parameter optimization on the speech data was not possible due to the small amount of data and because running a large number of speech recognition experiments would have been too time consuming. So a set of parameters that seemed to work fine was used. The bin estimation method has much more parameters than the other evaluated combination methods what for its parameters are not as easily adjusted for the new test data. For this reason, the parameters of the bin estimation method were not adjusted but used as they were learned from the text data. Due to the unreliable estimation of the parameters, the speech recognition results should be taken rather as suggestive than complete comparison of the methods.

4.4 Results

Perplexity results

The results of the perplexity experiments for the two model combinations are given in table 4.1. The relative perplexity reductions compared to the plain 4-gram model are presented in figure 4.1. The linear interpolation did not improve the result with the topic model but otherwise all combination methods yielded significant perplexity reduction over the plain 4-gram model that works here as a baseline. For all two-model combinations, the best performing method was the bin-method. The second best method, when applicable, was the unigram rescaling performing slightly worse than the bin-method. Notably worse was the log-linear interpolation achieving still 13 % improvement over the baseline. In all cases, the linear interpolation was the worst method producing only slight improvement over the baseline with the LSA

model and no improvement with the topic model. However, with the cache model the linear interpolation performed well producing nearly 25 % improvement. Also in this case, the bin estimation method was better achieving almost 32 % perplexity reduction. The simple 3-value cache model, combined with the 4-gram model by the bin-method, performed almost as well as the regular cache giving 30 % improvement. The results concerning the topic and the LSA model are quite similar to the results reported by Gildea and Hofmann (1999) and Coccaro and Jurafsky (1998). The exception is the linear interpolation that failed to produce any improvement when combining the topic model and the 4-gram model. This is discussed in section 4.6.

The results of the perplexity experiments for the three and four model combinations are presented in table 4.2. The relative perplexity reductions over the plain 4-gram model are depicted in figure 4.2. Following notation was used in presenting the combinations. In each pairwise combination the 4-gram works as the other component and is left out from the denotation. So "resc(topic)" refers to the combination where the 4-gram model and the topic model have been combined together using the unigram rescaling method. The only case where the "4-gram" has been written out is the combination 5 in which the 4-gram, topic, and the three-value cache model have all been combined together using the bin estimation method. The plus symbol (+) refers to linear interpolation. In the three and four model combinations, the pairwise combinations have been combined together using linear interpolation. The only exception is the combination 5 as explained above.

The smallest total perplexity was achieved by combination 5 where the 4-gram model, topic model, and the cache model are all combined together using the bin estimation method. This combination resulted in 46 % perplexity reduction which is one of the greatest reported perplexity reductions that has been achieved by combining language models. Almost equally good result, 44.8 %, was achieved by combination 3, in which the unigram rescaling combination of the 4-gram and the topic model was interpolated with the bin-method combination of the 4-gram and the cache model. For the two model combinations the best performing combination was resulted from the bin estimation method. It would be reasonable to assume, that when these two model combinations are further combined together using linear interpolation, the best performing combination would consist of these bin-method combinations. However, the best performance was achieved by combination 3, in which the topic model and the 4-gram model were combined together using unigram rescaling. Why this happens is unclear. The large test data should exclude the chance of statistical incident.

Adding the LSA model to the combination of the 4-gram model, topic model, and the cache model did only little further improvement to the result. The LSA and the topic model focus in modeling the same aspect of the language. Thus, only little cumulative improvement is achieved when combining these two together.

Table 4.1: Perplexity results for two model combinations

Model	Method	Perplexity
4-gram	-	5584
4-gram + LSA	linear	5495
4-gram + LSA	loglin	4828
4-gram + LSA	rescaling	4542
4-gram + LSA	bin-method	4428
4-gram + topic	linear	5584
4-gram + topic	loglin	4836
4-gram + topic	rescaling	3892
4-gram + topic	bin-method	3666
4-gram + cache	linear	4211
4-gram + cache	bin-method	3822
4-gram + cache	3-value-bin	3915

Table 4.2: Perplexity results for three and four model combinations

Combination	Perplexity
resc(4-gram + topic) lin cache	3634
bin(4-gram + topic) lin cache	3445
resc(4-gram + topic) lin bin(4-gram + cache)	3084
bin(4-gram + topic) lin bin(4-gram + cache)	3154
bin(4-gram + topic + 3-value-cache)	3009
resc(4-gram + topic) lin bin(4-gram + cache) lin resc(4-gram + LSA)	3071
resc(4-gram + topic) lin bin(4-gram + cache) lin bin(4-gram + LSA)	3068
bin(4-gram + topic) lin bin(4-gram + cache) lin resc(4-gram + LSA)	3142
bin(4-gram + topic) lin bin(4-gram + cache) lin bin(4-gram + LSA)	3150

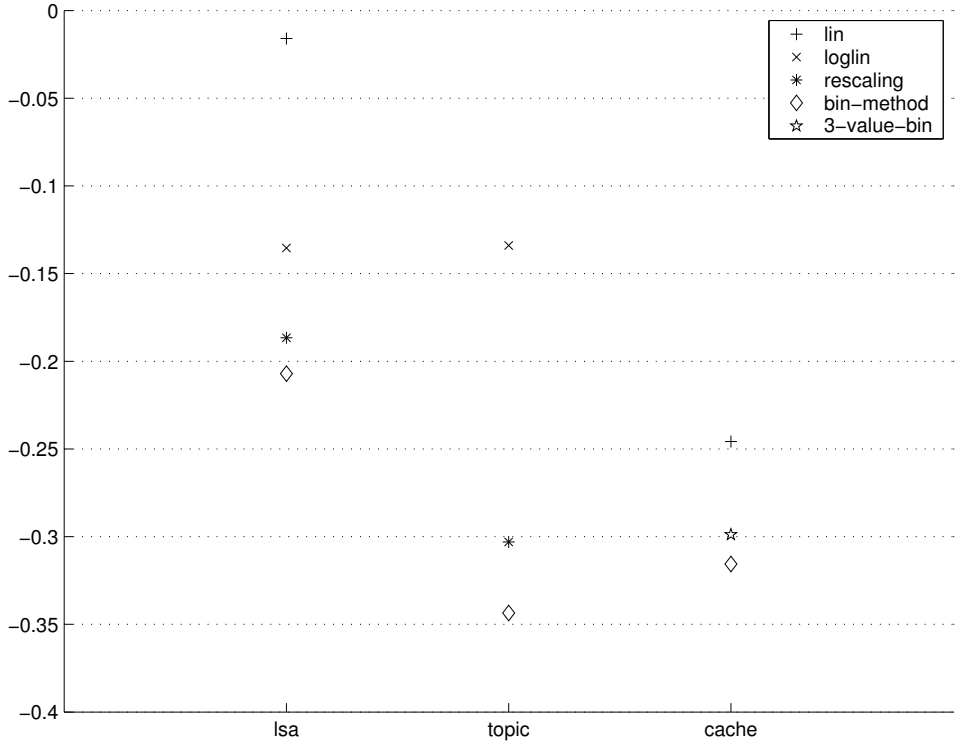

Figure 4.1: Relative perplexity reductions for two model combinations

To see how well the language models and different combinations performed on the transcription of the speech data used in the speech recognition experiments, perplexity experiments were run also for this data. The results are given in table 4.3 and the relative perplexity reductions are depicted in figure 4.3.

Speech recognition results

The results of the speech recognition experiments are given in table 4.4. The relative word error and phoneme error reductions to the baseline are depicted in figures 4.4 and 4.5. We see that the outstanding perplexity reductions turn into only negligible improvements in speech recognition results. While the bin-method was dominating in the perplexity experiments, such tendency is not observed in the speech recognition results. One reason for this is that the bin-method was not adjusted based on the speech data while the other methods were. So the results are not strictly comparable together. The best performing method varies depending on the model and also whether looking at the word or phoneme error rate. The significantly best performing combination was the unigram rescaling combination of the 4-gram and the topic model which achieved 4 % word error reduction and 7.5 % phoneme error reduction

Table 4.3: Perplexity results for the speech data

Model	Method	Perplexity
4-gram	-	12404
LSA	lin	12067
LSA	loglin	11320
LSA	rescaling	11323
LSA	bin-method	10662
topic	loglin	10551
topic	rescaling	9385
topic	bin-method	8789
cache	linear	8071
cache	bin-method	7263
cache	3-value-bin	7945
topic+cache	resc+bin+lin	7008
topic+cache	bin-method	6457

Table 4.4: Speech recognition results

Model	Method	WER (%)	PHER (%)
4-gram	-	25.70	5.87
LSA	linear	25.99	5.79
LSA	loglin	25.71	5.95
LSA	rescaling	26.12	5.92
LSA	bin-method	25.64	5.88
topic	loglin	26.27	6.08
topic	rescaling	24.66	5.43
topic	bin-method	25.28	5.54
cache	linear	26.57	5.91
cache	bin-method	27.76	5.88
cache	3-value-bin	25.29	5.73
topic+cache	resc+bin+lin	26.10	5.75
topic+cache	bin-method	24.71	5.50

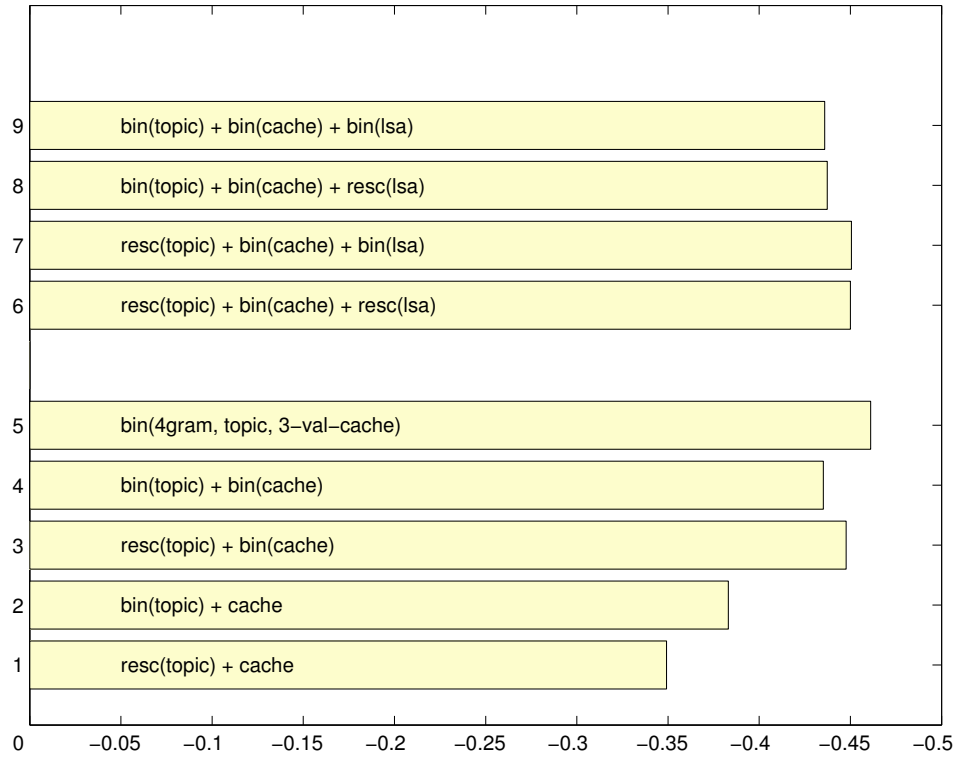

Figure 4.2: Relative perplexity reductions for three and four model combinations

compared to the plain 4-gram model.

Statistical analysis

Wilcoxon signed-rank test (Milton and Arnold, 1995) for paired observations was used to test the statistical significance of the results. The H_0 hypothesis was that the combined model and the baseline model perform equally well measured by the median of the document perplexities. All perplexity results were compared to the baseline and the H_0 hypothesis could be rejected with practically 100 % confidence for all combinations. The great level of confidence is due to the large test set and the Wilcoxon signed-rank test.

It seems that the bin-method is slightly but consistently better than the second best performing unigram rescaling method. The Wilcoxon signed-rank test shows that the conclusion is statistically justified as the H_0 hypothesis can be rejected again with practically 100 % confidence.

While the improvements in the perplexity results were clear, the improvements in

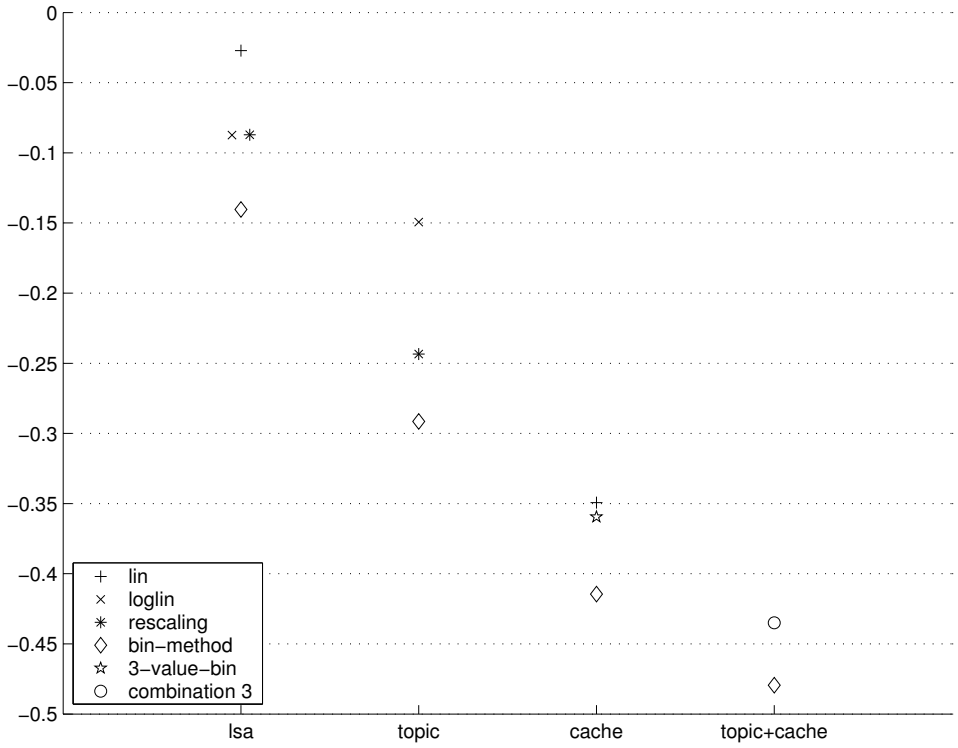

Figure 4.3: Relative perplexity reductions for the speech data

the speech recognition experiments were almost marginal. Measuring the word error rate, only the best performing combination, the topic model combined with the 4-gram model using the unigram rescaling method, proved to be statistically significant with confidence value of 98.8 %. Measuring the phoneme error rate, also the bin estimation method used with the topic model yielded improvement of statistical significance with confidence value of 99.93 %. The bin-method combination of the three models, 4-gram, topic, and the cache model, achieved almost as good results as the best performing combination. However, the confidence value calculated by the Wilcoxon test is 94 % when comparing the word error rates. So the improvement can not be considered statistically significant. Comparing the phoneme error rates the improvement is statistically significant. The improvement of any the other combination can not be stated statistically significant.

4.5 Normalization sum

A concession that we had to make in the speech recognition experiments was that the combined language model probabilities were left unnormalized. Bellegarda (2000) re-

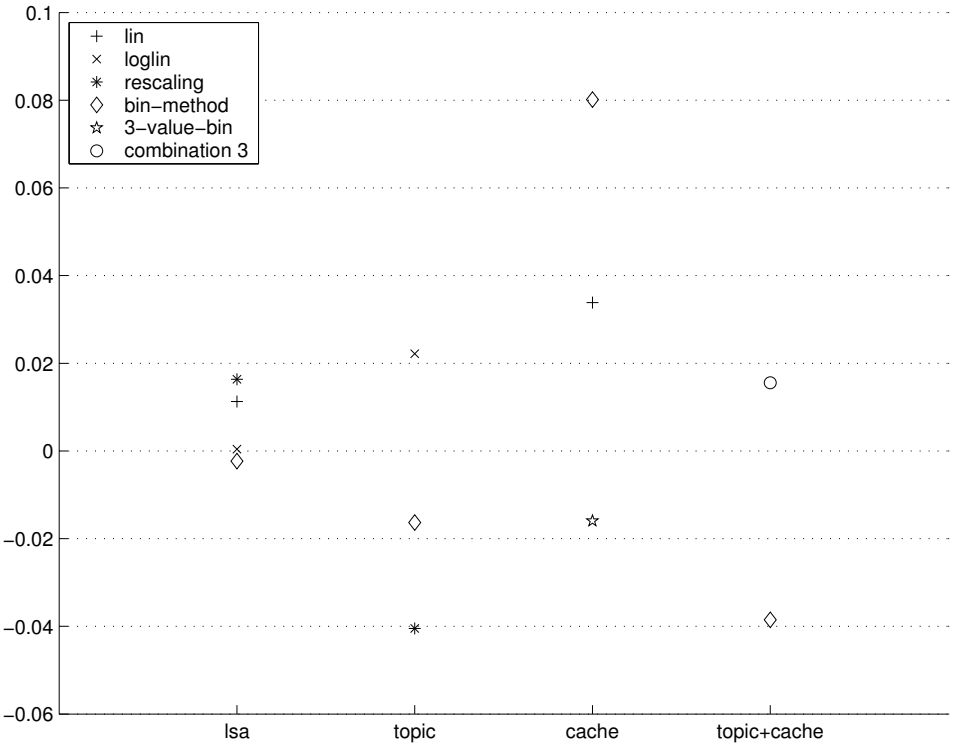

Figure 4.4: Relative word error rate reductions in speech recognition experiments

ports that when using n-gram model and the LSA model combined by the unigram rescaling method, no performance degradation was observed when making this approximation. However, it is interesting to see, do the combination methods differ in this aspect. The language model values not summing up to 1 is not dangerous itself, but if the sum varies a lot from time to time it is bound to have some consequences. One way to evaluate the combination methods is to measure the variance of the sum of the probabilities. If the variance is near to zero it implies that little distortion is introduced when the normalization is omitted. To measure the variance, the normalization coefficient, i.e. the sum of the probabilities of all words, was calculated for each time, and the mean and the variance for the coefficient were calculated. The results for the two-model combinations are given in table 4.5. The LSA model probabilities were also unnormalized themselves what for the probabilities do not sum to one even when using the linear interpolation. The outputs of the topic and the cache model are pure probabilities and so the linear interpolation has been left out from the table.

We see that when combining the LSA model and the 4-gram model the variances are quite small for all combination methods. When combining the topic model and the 4-gram model the unigram rescaling has much larger variance than the log-linear and

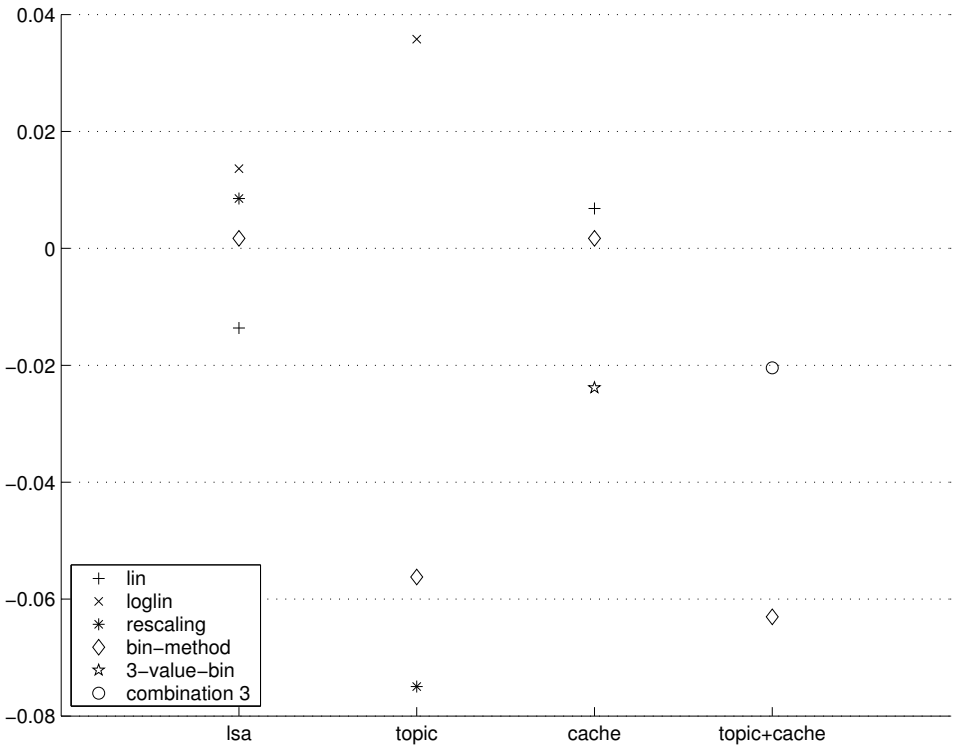

Figure 4.5: Relative phoneme error rate reductions in speech recognition experiments

Table 4.5: Normalization coefficient statistics

Model	Method	Mean	Variance
4-gram + LSA	linear	12	18
4-gram + LSA	loglin	0.29	0.0088
4-gram + LSA	rescaling	0.50	0.047
4-gram + LSA	bin-method	0.99	0.037
4-gram + topic	loglin	0.45	0.062
4-gram + topic	rescaling	0.81	0.98
4-gram + topic	bin-method	0.98	0.016
4-gram + cache	bin-method	1.00	0.026
4-gram + cache	3-value-bin	0.98	0.040

the bin estimation methods. This suggests that the bin estimation method and the log-linear interpolation should suffer even less than the unigram rescaling method from omitting the normalization.

4.6 Analysis

Measured by the perplexity, the bin estimation method performed best in all three cases of combining the 4-gram model with one of the large context models. It seems that the bin estimation method succeeds in its task of estimating the likelihood defined in equation 3.15. Performing best in all cases, of which two were of quite different nature, suggests that the method may be applicable with different kind of models.

The margin between the bin-method and the unigram rescaling is larger with the topic model than with the LSA model. This is probably due to the different pre-processing of the component model probabilities when using the bin-method. The topic model outputs were divided by the unigram probability of the particular word, whereas the LSA outputs were not.

To understand more deeply how the bin method works let us look inside what the method has learned. In figure 4.6 the likelihood values attached to each bin for combining the 4-gram model and the topic model are presented. We see that the 4-gram model is in a dominating role while the topic model seems to have effect mainly with very big or small values. In figure 4.7 horizontal cross-sections with different 4-gram probabilities are plotted from the figure 4.6. In each curve the 4-gram probability is kept constant and the topic model factor is swept from 0 to the maximum value[1]. To make the curves fit nicely into the same picture, each curve is scaled by dividing its values by their median. This way the curves cross in the point where the topic factor value is close to 1.0 which means that the topic model probability for the word is equal to its global unigram probability. When moving to the right from this point, we see how the increase in the topic factor value affects the likelihood value. Correspondingly, moving to the left shows how the likelihood changes when the topic factor value is decreased. From the plot we see that when the 4-gram probability is high the topic value has almost no effect to the output value. With the lower 4-gram probability values the topic model has more and more contribution and the corresponding curve approaches the reference curve which depicts the case of multiplying the probability values together as is done in the unigram rescaling. Actually, this plot does not tell the whole truth since the probability for a word depends also from the values of all other words through the normalization.

From the figure 4.6 one can make an intuitive deduction that a simple method, which could possibly work in combining the n-gram model and the topic model, would be

[1]The factor does not have a defined upper limit so all values above 26.2 in this case are mapped to the rightmost bins.

to use formula

$$P(w|h) \sim P_N(w|h) \left(\frac{P_T(w|h)}{P_{unigram}(w)} \right)^{\lambda(P_N(w|h))} \tag{4.1}$$

where the exponent $\gamma(P_N)$ is adjusted by the n-gram probability $P_N(w|h)$. To try this is left for future work.

Two reasons can be given for the success of the bin-method. First, no assumptions are made about the models that are combined. The models can be independent or strongly correlated. Second, the only assumption that is made about the estimated likelihood $p(w = w_i|P_A, P_B)$ is that it is piecewise constant. In the linear interpolation, log-linear interpolation, and the unigram rescaling, the likelihood is modeled by a predefined function that has a few parameters that are optimized on data. The bin estimation method, like histogram methods generally, allows more freedom to the function that is estimated. On the other hand, the cost of the flexibility is the increased amount of data that is needed to reliably train the method.

With the LSA and the topic model the second best performing method after the bin estimation method was the unigram rescaling. The log-linear interpolation performed notably worse achieving still significant ($\sim$13 %) improvement to the baseline. There is a great difference in the performance of the log-linear interpolation and the unigram rescaling when combining the topic model and the 4-gram model but the difference is much smaller when combining the LSA model and the 4-gram model. A probable reason for this is following. The LSA model and the topic model are very similar. Both of them try to measure the semantic closeness of the words and the current context. The topic model can be seen to model the topic dependent fluctuation of the marginal distribution of the words. By dividing the probability estimate of the topic model for word w_i by the unigram probability of the word $P_{uni}(w)$, as is done in the unigram rescaling method, we get a factor that describes the relative frequency of the word compared to the its global unigram probability. In the construction of the co-occurrence matrix in the LSA model, the frequent words that occur in several documents are punished using the idf-weighting. Thus the output of the LSA model is neither a pure marginal probability nor a relative frequency factor. For this reason, the optimal scaling exponents are quite different for the LSA model than for the topic model, and the difference in the results of the log-linear and the unigram rescaling methods is much smaller with the LSA model than with the topic model.

In combining the LSA and the 4-gram model, the linear interpolation yielded only small perplexity reduction. The result is in line with the previously reported results of Coccaro and Jurafsky (1998). In combining the topic model and the 4-gram model, the linear interpolation failed to produce any improvement over the baseline. This is an unexpected result since Gildea and Hofmann (1999) report notable improvement over the baseline when combining the topic model with the trigram model using the linear interpolation. The reason behind the bad performance may be the word borders that are also predicted by the topic model. The 4-gram model manages to

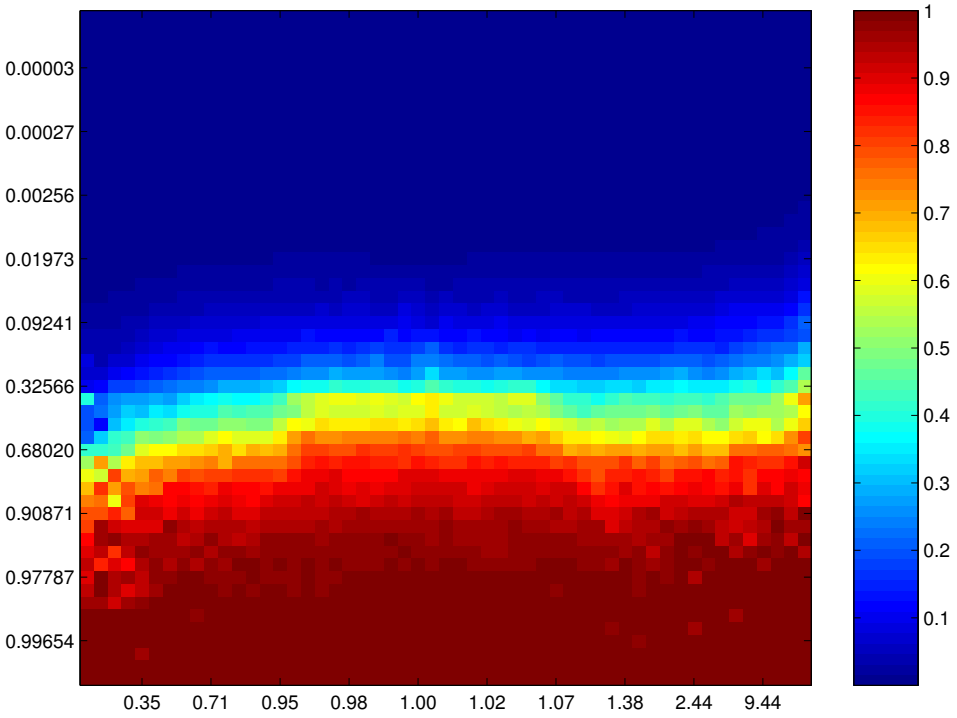

Figure 4.6: Bin-method likelihood table for the topic model

predict the word borders with good accuracy, and interpolating it with the topic model will in almost all cases reduce the probability estimate of the word border. As the word borders constitute over one third of all tokens, the worsened accuracy in predicting them degrades the total performance of the model. In the LSA model the word borders were not included but the probability for the word borders was determined merely by the 4-gram model.

While in combining the LSA and the topic model with the 4-gram model the linear interpolation performed bad, it performed comparably well in combining the cache model and the 4-gram model. Looking at the perplexity results for the three and four model combinations in figure 4.2, it is seen that interpolating the two-model combinations together yielded significant improvement to the two model results. Also in the experiments of Goodman (2000), remarkable perplexity reductions were achieved by using the linear interpolation in combining several language models. To understand why the linear interpolation performs well in some cases while in some cases it is almost useless, we have to remember that the averaging that is done in the linear interpolation is inclined to smooth the output distribution. This is particularly the case when one of the component models is significantly smoother than the others. In some cases the smoothing may reduce the total perplexity by raising the low probability estimates. However, the averaging is bound to worsen the high probability

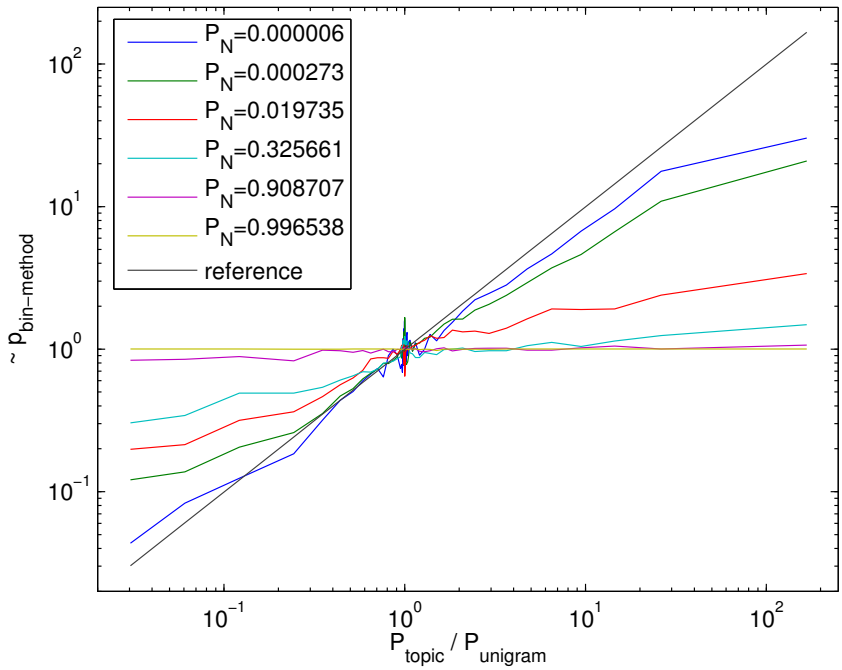

Figure 4.7: Relative likelihoods for the combination of the topic model and the 4-gram model with different 4-gram values. The curves are median scaled.

estimates. My conclusion is that the linear interpolation can be efficient when any of the component models is not significantly smoother than the other models. This is the case when the models incorporate the same amount of local information, for example, in the case of n-gram-based models of approximately same order. When one of the component models is significantly smoother than the others, the damage produced by the smoothing becomes greater than the gain that is possibly brought in by the model.

Chapter 5

Conclusions

In this work several methods for combining language models have been studied. The purpose was to give the reader a picture of available methods and their properties, what are the pros and cons for each method and what kind of situations do they suit for. In addition, a new combination method, called bin estimation method, was presented.

A theoretically justified way of combining different information sources is provided by the maximum entropy modeling. In previous work the ME modeling has been successfully used with a variety of models showing that the approach has also practical potential. Practical issues however have restricted the use of the method, as the training and the use of the ME models are computationally demanding. However, the vivid research and the use of efficient algorithms have brought the method to one option when choosing the combination method for language models.

In many cases, sufficient performance can be achieved by more simple methods than the ME modeling. For example, in the case of combining the topic or the LSA model with the n-gram model, a simple formula, called unigram rescaling, has been drawn for the probability under relatively mild assumptions.

In this work, four combination methods: linear interpolation, log-linear interpolation, unigram rescaling, and bin estimation method were evaluated by perplexity and speech recognition experiments. In the perplexity experiments, the best performing method was the bin estimation. The greatest overall perplexity reduction, 46 %, was achieved by using the bin estimation method in combining 4-gram, topic, and the cache models together. The result is one of the greatest perplexity reductions that has been reported over the properly smoothed 4-gram model. However, the remarkable perplexity reductions turned only into small improvements in the speech recognition experiments.

The presented bin estimation method is one possible implementation for using multivariate function estimation methods in the task of combining language models. It is left for future work to study whether improvements could be achieved by more thorough parameter optimization or using some other function estimation methods. In the bin estimation method, no assumptions are made about the models to be combined. Also, the only restriction set to the likelihood function is that it has to be slowly variating enough to be accurately estimated by the histograms. These things and the better performance compared to the other evaluated methods suggest that the bin-method may be applicable in combining many kind of models. However, to make further conclusions would need more experiments with different kind of language models.

One possible application of the method is to use it in analyzing the joint behavior of the models. That way it may be possible to find more simple combination methods for different models. The disadvantage of the bin method is that it needs a large amount of data to train the combination parameters properly. This may restrict the number of the models that can be combined simultaneously with the method.

Probably the most commonly used combination methods, the linear interpolation and the backoff method, have still an important role in language modeling in combining different models. The linear interpolation is fast to calculate and the parameter estimation is easy as it has very little parameters and the heavy calculation of the normalization is avoided. In cases where the linear interpolation does not introduce severe averaging, it seems to have relatively good performance and it may be a good choice for the combination method. The backoff method has shown to perform worse than the linear interpolation method when used in combining different order n-gram models. Whether this is a common rule is still uncertain.

Some interesting methods, like maximum entropy approach, were considered only on theoretical level. To go deeper in the subject, more experiments should be run using different language models and data sets.

Bibliography

Bahl, L., Brown, P., de Souza, P., Mercer, R., 1989. A tree-based statistical language model for natural language speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (7).

Bellegarda, J., 1997. A latent semantic analysis framework for large-span language modeling. In: Eurospeech.

Bellegarda, J., Butzburger, J., Y. Chow, N. C., Naik, D., 1996. A novel word clustering algorithm based on latent semantic analysis. In: Proceedings of ICASSP-96.

Bellegarda, J. R., August 2000. Exploiting latent semantic information in statistical language modeling. In: Proceedings of the IEEE. Vol. 88. IEEE, pp. 1279–1296.

Berger, A. L., Della Pietra, S. A., Della Pietra, V. J., 1996. A maximum entropy approach to natural language processing. Computational Linguistics 22 (1), 39–71.

Chen, S. F., Beeferman, D., Rosenfeld, R., 1998a. Evaluation metrics for language models. In: DARPA Broadcast News Transcription and Understanding Workshop. pp. 275–280.

Chen, S. F., Goodman, J., 1998. An empirical study of smoothing techniques for language modeling. Tech. rep., Harvard University.

Chen, S. F., Rosenfeld, R., January 2000. A survey of smoothing techniques for me models. IEEE Transactions on Speech and Audio Processing 8 (1).

Chen, S. F., Seymore, K., Rosenfeld, R., 1998b. Topic adaptation for language modeling using unnormalised exponential models. In: ICASSP-98. Seattle, Washington.

Clarkson, P. R., Robinson, T., 1997. Language model adaptation using mixtures and an exponentially decaying cache. In: IEEE ICASSP-97. Munich, Germany, pp. 799–802.

Coccaro, N., Jurafsky, D., November 1998. Towards better integration of semantic predictors in statistical language modeling. In: ICSLP. ICSLP, Sydney, Australia.

Creutz, M., Lagus, K., July 2002. Unsupervised discovery of morphemes. In: Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02. Philadelphia, Pennsylvania, pp. 21–30.

Csiszàr, I., 1975. I-divergence geometry of probability distributions and minimization problems. Annals of Probability 3 (1), 146–158.

Darroch, J., Ratcliff, D., 1972. Generalized iterative scaling for log-linear models. The Annals of Mathematical Statistics 43, 1470–1480.

Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W., Harshman, R. A., 1990. Indexing by latent semantic analysis. Journal of the Society for Information Science 41 (6), 391–407.

Della Pietra, S., Della Pietra, V., Lafferty, J., 1997. Inducing features of random fields. Transactions on Pattern Analysis and Machine Intelligence 19 (4), 280–393.

Geutner, P., 1995. Using morphology towards better large-vocabulary speech recognition systems. In: Proceedings of ICASSP. pp. 445–448.

Gildea, D., Hofmann, T., 1999. Topic-based language modeling using em. In: Proc. Eurospeech. pp. 2167–2170.

Good, I., 1953. The population frequencies of species and the estimation of population parameters. Biometrika 40 (3/4), 237–264.

Goodman, J. T., June 2000. Putting it all together: Language model combination. In: ICASSP'00. Vol. 3. IEEE, pp. 1647–1650.

Goodman, J. T., 2001. A bit of progress in language modeling. Tech. rep., Microsoft Research.

Hirsimäki, T., 2002. A decoder for large vocabulary continuous speech recognition. Master's thesis, Helsinki University of Technology.

Huang, X., Alleva, F., Hon, H., Hwang, M., Lee, K., Rosenfeld, R., 1993. The sphinx-ii speech recognition system: An overview. Computer, Speech, and Language 2, 137–148.

Iyer, R. M., Ostendorf, M., January 1999. Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. IEEE Transactions on Speech and Audio Processing 7 (1).

Jaines, E., May 1957. Information theory and statistical mechanics. Physics Reviews (106), 620–630.

Jelinek, F., Mercer, R., 1980. Interpolated Estimation of Markov Source Parameters from Sparse Data. In Pattern Recognition in Practise. North Holland, Amsterdam.

Jelinek, F., Merialdo, B., Roukos, S., Strauss, M., 1991. A dynamic lm for speech recognition. In: Proc. ARPA Workshop on Speech and Natural Language. pp. 293–295.

Kalai, A., Chen, S., Blum, A., Rosenfeld, R., 1999. On-line algorithms for combining language models. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing.

Katz, S. M., March 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech, and Signal Processing 35 (3), 400–401.

Kirchhoff, K., Bilmes, J., Henderson, J., Schwartz, R., Noamany, M., Schone, P., Ji, G., Das, S., Egan, M., He, F., Vergyri, D., Liu, D., Duta, N., 2002. Novel speech recognition models for arabic. Tech. rep., Johns-Hopkins University Summer Research Workshop.

Kneser, R., Ney, H., 1995. Improved backing-off for m-gram language modeling. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Vol. 1. pp. 181–184.

Kuhn, R., de Mori, R., June 1990. A cache-based natural language model for speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (6).

Kurimo, M., Lagus, K., 2002. An efficiently focusing large vocabulary language model. In: Conference on Artificial Neural Networks (ICANN'02). pp. 1068–1073.

Malouf, R., 2002. A comparison of algorithms for maximum entropy parameter estimation. In: 6th Conference on Natural Language Learning. SIGNLL.

Martin, S., Hamacher, C., Liermann, J., Wessel, F., Ney, H., September 1999. Assesment of smoothing methods and complex stochastic language modeling. In: 6th European Conference on Speech Communication and Technology. Vol. 5. Budapest, Hungary, pp. 1939–1942.

Martin, S., Liermann, J., Ney, H., 1997. Adaptive topic-dependent language modeling using word-based varigrams. In: Proceedings of Eurospeech.

Milton, J., Arnold, J. C., 1995. Introduction to probability and statistics, third edition Edition. McGraw-Hill international editions.

Ney, H., Essen, U., Kneser, R., 1994. On structuring probabilistic dependences in stochastic language modeling. Computer Speech and Language 8, 1–38.

Pylkkönen, J., 2004. Phone duration modeling techniques in continuous speech recognition. Master's thesis, Helsinki University of Technology.

Rosenfeld, R., April 1994. Adaptive statistical language modeling: A maximum entropy approach. Ph.D. thesis, Carnegie Mellon University.

Schultz, T., Kiecza, D., Waibel, A., 1999. Data-driven determination of appropriate dictionary units for korean lvcsr. In: Proceedings of ICASSP.

Siivola, V., Hirsimäki, T., Creutz, M., Kurimo, M., September 2003. Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner. In: Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech). Geneva, Switzerland, pp. 2293–2296.

Silverman, B., 1986. Density Estimation for Statistics and Data Analysis. Chapman & Hall, London.

Siu, M., Ostendorf, M., 2000. Variable n-grams and extensions for conversational speech language modeling. IEEE Transactions on Speech and Audio Processing 8, 63–75.

Witten, I., Bell, T., July 1991. The zero frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory 37, 1085–1094.

Wu, J., 2002. Maximum entropy language modeling with non-local dependencies. Ph.D. thesis, Johns Hopkins University, Baltimore, Maryland.