



UNIVERSITY OF
CAMBRIDGE

Churchill College

MPhil Computer Speech and Language Processing

Minimum Word Error Rate Decoding

Gunnar Evermann

August 1999

Abstract

From the theory of Bayesian pattern recognition it is well known that the *maximum a posteriori* decision criterion yields a recogniser with the minimum probability of assigning the incorrect label to a pattern, if the correct probability distributions are used. This MAP criterion is also routinely employed in automatic speech recognition system. The problem addressed in this thesis is the fact that in a continuous speech recognition task the MAP criterion only guarantees to minimise the sentence error rate, but for many application the by far more relevant error metric is the *word* error rate.

In the general framework of lattice rescoring an approach is investigated that tries to directly use the word error metric in the objective function that is the basis of the decision in the recogniser. An implementation of two variants of this approach is presented and its performance is evaluated on two very different continuous speech corpora.

Closely related to this is the problem of replacing the *a posteriori* probabilities based on whole sentences that are used in conventional decoders by posterior probabilities on the word level. A new technique of estimating these word posteriors is presented and applied with the aim of improving the decoding result and also providing accurate confidence scores.

Declaration

I declare that this thesis is substantially my own work. Where reference is made to the works of others the extent to which that work has been used is indicated and duly acknowledged in the text and bibliography.

This thesis does not exceed 15000 words in length, including footnotes, bibliography and appendices.

Gunnar Evermann

Cambridge, 31.8.1999

Contents

1	Introduction	7
2	Speech Recognition Basics	9
2.1	Pattern Classification	9
2.2	Recognising Isolated Words	10
2.3	Continuous Speech	12
2.4	Decoding	13
2.5	Word Lattices	15
2.6	Evaluation procedure	16
3	Posterior Probabilities	19
3.1	Sentence vs. Word Based Metrics	19
3.2	Word-level Posterior Probabilities	22
3.2.1	Keyword Spotting	22
3.2.2	Confidence Scores	24
3.2.3	Definition	25
3.3	Calculating Posterior Probabilities in Lattices	26
3.4	Time Dependent Word Posteriors	29
4	Minimising Word Error Rate	33
4.1	Expected Word Error Rate	33
4.2	N-best List based Minimisation	34
4.3	Confusion Networks	38
5	Experimental Results	43
5.1	Description of Corpora	43
5.2	Lattice Pruning	44
5.3	Decoding	46
5.3.1	N-best List Rescoring	46
5.3.2	Time Dependent Posteriors	48
5.3.3	Confusion Networks	49
5.3.4	Summary	49

Contents

5.4 Confidence Scores	50
6 Conclusions & Further Work	53
Bibliography	55

1 Introduction

In the last two decades significant progress has been made in the field of automatic speech recognition. The state of the art has progressed from simple speaker dependent isolated speech recognisers to systems capable of recognising conversational dialogues with vocabularies of over 50,000 words. For a long time speech recognition systems were only designed as research platforms but the advances of the underlying technology now make it possible to develop systems that solve specific real-world problems and can be deployed as successful products. The most common examples of such products available today are large vocabulary dictation systems and automated (telephone) dialogue systems (typically used for tasks such as timetable inquiries or booking cinema tickets). In the near future it is to be expected that more ambitious products will be developed in which speech recognition is only one component integrated in a complex natural language processing system. Examples of this are speech-to-speech machine translation systems which are already being designed and built for restricted domains in many research labs.

In the past speech recognition systems have usually been developed and optimised in isolation, i.e. no special provisions were made to ensure that the recogniser was optimally tuned for the actual task at hand. Different applications might very well impose different requirements on the recogniser. Quite often these specific requirements are ignored and the recogniser is just optimised with respect to some standard criterion.

The overall approach to the speech recognition problems has been largely unchanged for a long time. Probabilistic models are trained to capture the variability of speech. The actual recognition is performed by finding the word string that yields the maximum a-posteriori probability given the trained models and the observed speech. This is a straightforward approach as it is based on well understood principles of Bayesian pattern recognition, where it indeed represents the optimal classification criterion. The problem with the application of the MAP approach to speech recognition is that it is not optimal with respect to minimising the number of word errors in the system output.

In this thesis some alternatives to the MAP decoding criterion are examined which try to find a solution to the decoding problem that directly minimises the word error rate. They rely on some form of word level criterion (instead of the sentence level MAP measure). These techniques are closely related to a number of other research areas in the field of speech recognition, such as keyword spotting and the estimation of confidence scores.

In the following chapter a short overview of the statistical approach to speech recognition is given with a special emphasis on treating speech recognition as an instance of the more general class of pattern classification problems. The concept of *a posteriori* probabilities at different levels (i.e. sentence or word level) is central to many improved decoding methods and will be discussed in chapter 3. A method to estimate these posterior probabilities at the word level from the output of a conventional MAP decoder is presented. In addition the use of word posteriors as measures of confidence is illustrated. In chapter 4 the approach of minimum word error rate decoding is described and possible implementations based on rescoreing the output of a conventional MAP decoder are discussed. The performance of this approach in comparison to the traditional decoding is examined in chapter 5 where experimental recognition results on two different continuous speech corpora are presented. General conclusions are drawn in chapter 6 and areas for further research are identified.

2 Speech Recognition Basics

In this chapter some of the theoretical foundations of speech recognition in the general framework of pattern classification are discussed. First an overview of the pattern classification problem and the statistical approach to solving it is given. Then the specific algorithms employed in the development of speech recognition systems are described with special emphasis on the actual recognition and the evaluation of the quality of the recognition results.

2.1 Pattern Classification

The process of *Pattern Classification* is of fundamental importance to many of the problems tackled in the field of Artificial Intelligence research. A rich set of techniques has been developed and applied to a variety of problems. In the following an overview of the basic approach in *statistical (Bayesian)* pattern recognition is given.

It is assumed that each pattern belongs to exactly one class ω out of a (potentially infinite) set of classes $\Omega = \{\omega_1, \omega_2, \dots, \omega_c\}$. A pattern is characterised by a set of measurements¹ (called *features*). These features are combined to form a feature vector $\mathbf{x} = (x_1, x_2, \dots, x_d)$. It should be noted that the correspondence between feature vectors and classes need not be deterministic, i.e. different classes can result in the same feature vector being observed. Therefore both \mathbf{x} and ω are treated as random variables.

The process of pattern classification is achieved by a mapping from a feature vector to the corresponding class. This mapping is encoded in a decision function $g : \mathbf{R}^d \rightarrow \Omega$. The objective in the development of a classifier is to find a decision function that is optimal in the sense that the number of misclassifications is reduced to a minimum.

The misclassification rate is defined as the probability of assigning a feature vector to the wrong class:

$$L(g) = P(g(\mathbf{x}) \neq \omega(\mathbf{x})) \quad (2.1)$$

¹The kind of measurements used depends on the nature of the application domain and the observed patterns. They can be as diverse as gray scale intensities in an image processing context and stock prices in a financial application

Due to the non-deterministic nature of the mapping between feature vectors and classes there is actually a lower bound on the misclassification probability that can be achieved. This bound is commonly called the *Bayes Error* (L^*) and the corresponding decision function the *Bayes Classifier* (g^*).

To formalise these notions the process of pattern generation is viewed as a two-stage stochastic process. In the first stage a class ω_i is chosen (the associated probability distribution $P(\omega)$ is called the *a-priori* distribution). The second stage is the generation of the actual pattern according to the distribution $p(\mathbf{x}|\omega_i)$ which depends on the previously chosen class.

It is well known that the optimal classifier (i.e. the one with the smallest probability of misclassification) is defined by (see e.g. [Duda and Hart, 1973]):

$$g^*(\mathbf{x}) = \operatorname{argmax}_i P(\omega_i|\mathbf{x}) \quad (2.2)$$

This decision function picks the class ω_i with the highest *a-posteriori* probability. Therefore this decision criterion is also called *Maximum A-Posteriori* (MAP) criterion. As these probabilities are generally unknown they are decomposed according to *Bayes Rule*:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \quad (2.3)$$

yielding:

$$g^*(\mathbf{x}) = \operatorname{argmax}_i \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} = \operatorname{argmax}_i p(\mathbf{x}|\omega_i)P(\omega_i) \quad (2.4)$$

where the probability of the observed feature vector $p(\mathbf{x})$ is ignored as it is independent of the class ω_i and therefore does not have any influence on the decision.

In a statistical pattern classifier models of the the prior distribution $p(\omega_i)$ and the class dependent densities $p(\mathbf{x}|\omega_i)$ are estimated from a set of labelled training samples $\chi = \{(\mathbf{x}_1, \omega_1), \dots, (\mathbf{x}_N, \omega_N)\}$. The most common approach is to use the probabilities from these models in place of the true probabilities in equation 2.4 (*plug-in method*).

2.2 Recognising Isolated Words

The general approach to pattern classification can be directly applied to automatic speech recognition. Here the objective is to recognise the words spoken in an utterance. The difficulty of a speech recognition task depends on various factors such as the size of the vocabulary, the style of speech and the recording conditions. The following gives

a very brief overview of the statistical approach to speech recognition. For a more in-depth description see [Rabiner and Juang, 1993] and [Schukat-Talamazzini, 1995] which provides a good description of many more advanced techniques.

Historically the first speech recognition systems were designed to recognise words one at a time, i.e. either the domain was restricted to single word utterances (yes/no or single digits) or the words had to be spoken with a short pause in between them to allow an automatic system to segment the utterance into words which were then recognised separately (isolated speech). In this case the application of the Bayesian pattern recognition approach is straightforward. The set of classes Ω to be recognised is the vocabulary of the system and each class ω_i corresponds to a different word.

The speech signal is converted into a feature representation by segmenting it into short *frames* (typically around 20ms in length) whose spectral properties are encoded in a low dimensional feature vector \mathbf{x} (the most widely used representation are *Mel Frequency Cepstral Coefficients* typically yielding 39 values per frame). This sequence of features $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_T$ is used as the input of the statistical model.

In this notation the MAP decision rule takes the following form:

$$\hat{w} = \operatorname{argmax}_{w \in \mathcal{V}} p(\mathbf{X}|w)P(w) \quad (2.5)$$

where \mathcal{V} is the vocabulary and \hat{w} is the recognition result picked by the classifier.

Using this rule ensures that the probability of misclassification (i.e. the expected word error rate) is minimised. As the actual probabilities on the right hand side of equation 2.5 are in general unknown they are approximated by estimates based on the training corpus. The prior probabilities $P(w)$ can relatively easily be derived from simple frequency counts of the words in the corpus.

The probability densities $p(\mathbf{X}|w)$ are much more difficult to estimate robustly. To make this estimation feasible at all, it is assumed that the distribution comes from a family of distributions with a fixed set of parameters. The assumption most commonly used is that the sequence of feature vectors is generated by a first order Markov process. As the states of this underlying process cannot be observed directly this model is called a *Hidden Markov Model* (HMM). Typically these models have a small number of states (e.g. 15 per word) and only transitions to the following state and self-transitions are allowed.

With each state of the model an *output* probability density function is associated that captures the probability distribution of feature vectors produced in this state. Typically mixture Gaussian distributions are used. This model is used to assign probabilities to a given signal \mathbf{X} . This is achieved by calculating the probability of the signal for each possible path $\mathbf{q} \in \mathcal{Q}^T$ (i.e. sequence of states) through the HMM and summing over all these paths:

$$p(\mathbf{X}|w) = \sum_q p(\mathbf{X}, \mathbf{q}|w) \quad (2.6)$$

By introducing the HMM assumption the problem of estimating the probabilities in equation 2.5 is reduced to estimating a fixed number of parameters (the transition probabilities and the parameters of the output densities, e.g. means and variances of the normal distributions). These parameters are usually chosen so that the likelihood of the training data given the model set is maximised (*Maximum Likelihood Estimation*).

This estimation problem can be solved efficiently by the *Baum-Welch* algorithm (also called *Forward-Backward* algorithm) which is an instance of the class of iterative *Expectation Maximisation* algorithms and is guaranteed to increase the likelihood of the training data in each iteration (until convergence when the likelihood remains constant).

2.3 Continuous Speech

For many applications it is desirable (or even necessary) to be able to recognise normal continuous speech (i.e. without artificial pauses between words). It is well known that it is very hard to detect word boundaries in continuous speech (this is also true for human listeners and becomes more obvious when one tries to mentally segment sentences in a foreign language into words). For this reason it was necessary to develop techniques to recognise words without relying on a prior segmentation.

The speech signal is typically only segmented at long periods of silence (corresponding to for example sentence boundaries, hesitations or changes of speaker) yielding segments of up to one minute in length (see for example [Woodland et al., 1998b]). These segments are called utterances or sentences (which is somewhat inaccurate as they do not necessarily correspond to any kind of linguistically meaningful units) and are fed directly into the recognition process.

Now the basic classes which have to be distinguished by the pattern classifier are *sequences of words* (again often called sentences). It is *a-priori* not even known how many words this sequence contains. Therefore the probabilities of all possible words sequences over the vocabulary have to be considered. The MAP decision rule becomes:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} p(\mathbf{X}|\mathbf{W})P(\mathbf{W}) \quad (2.7)$$

where \mathbf{W} is a word sequence ($\mathbf{W} = w_1, \dots, w_n$) and $\hat{\mathbf{W}}$ is the recognition result.

The two unknown distributions are generally referred to as the *acoustic model* and the *language model* respectively.

The requirement of recognising continuous speech is often coupled with an increased vocabulary size. In this case it is usually no longer possible to train specific HMMs

for each word as the amount of training data is limited and many words occur only very seldomly. Instead words are decomposed into subword units (typically phones) according to a pronunciation dictionary and the parameters of these models are trained from the data. Thus the data is used more effectively and even models for words that do not occur in the training corpus at all can be synthesised by concatenating the HMMs of the corresponding subword units.

The estimation of the language model is also far more complex than in the isolated word case. Probabilities for all possible sequences of words in the vocabulary have to be estimated. The probability $P(\mathbf{W})$ is factored as follows:

$$P(\mathbf{W}) = P(w_1, \dots, w_n) \quad (2.8)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_n|w_1w_2 \dots w_{n-1}) \quad (2.9)$$

These conditional probabilities can often still not be estimated robustly from a limited training set. Therefore it is convenient to assume that only a fixed number of previous words have an influence on the probability of the current word (this is equivalent to assuming that the underlying process is a Nth order Markov process):

$$P(w_i|w_1w_2 \dots w_{i-1}) \approx P(w_i|w_{i-N+1} \dots w_{i-1}) \quad (2.10)$$

The resulting model is called a N-gram language model. Various estimation procedures and refinements are described in [Schukat-Talamazzini, 1995].

2.4 Decoding

The process of recognising the spoken words from the acoustic signal is often called *decoding* in reference to the information-theoretic channel model of speech production and recognition. The aim is to find the sequence of words spoken based on the feature vector representation of the speech signal and the set of parameters defining the acoustic and language model, which were optimised in the training phase.

The decoder tries to find the word sequence that is optimal according to an *objective function* (also *decoding function*). If the MAP criterion discussed in section 2.1 is adopted then the objective function is just $p(\mathbf{W}|\mathbf{X})$. As described in section 2.2 the calculation of this probability involves a summation over all possible state sequences (paths) through the HMMs corresponding to \mathbf{W} . This would be computationally very expensive and therefore a simplifying assumption is made:

$$p(\mathbf{X}|\mathbf{W}) = \sum_{\mathbf{q}} p(\mathbf{X}, \mathbf{q}|\mathbf{W}) \quad (2.11)$$

$$\approx \max_{\mathbf{q}} p(\mathbf{X}, \mathbf{q}|\mathbf{W}) \quad (2.12)$$

The assumption that the likelihood of the best path $\hat{\mathbf{q}}$ adequately approximates the sum over all paths is known as the *Viterbi approximation*. This significantly simplifies the decoding process as the problem can now be solved by just finding the best path through the search network that is generated by connecting the word HMMs appropriately. This path $\hat{\mathbf{q}}$ with the highest likelihood corresponds to a unique word sequence $\hat{\mathbf{W}}$. The Viterbi approximation is often quite acceptable as the likelihood of the best path usually indeed dominates the sum over all likelihoods.

An added complication is the fact that the two factors used in the likelihood calculation have very different dynamic scales. If they are just multiplied as indicated in equation 2.7 the decision for a word sequence would be dominated by the acoustic likelihoods and the language model would have hardly any influence. To balance the likelihoods it is customary to use an exponential weighting factor for the language model (*grammar scale* γ). The difference in dynamic scales is mainly caused by the fact that the acoustic likelihoods are severely underestimated as consecutive frames are assumed to be independent in the estimation of the HMM output densities.

As the effect of this underestimation is much less severe at word boundaries the decoder tends to select sentences consisting of many short words. This effect is compensated by introducing the so called *word insertion penalty* ρ which is added at the end of every hypothesised word. Incorporating these two extra factors the MAP decoding function becomes:

$$f(\mathbf{W}) = p(\mathbf{X}, \hat{\mathbf{q}}|\mathbf{W})P(\mathbf{W})\gamma\rho^{\|\mathbf{W}\|} \quad (2.13)$$

A more detailed discussion of the grammar scale and word penalty factors can be found in [Schukat-Talamazzini, 1995]. An advantage of these two factors is that they can be incorporated in the search network as extra weights on the appropriate edges.

The task to find the optimal path through the search network can be solved in a variety of different ways. A major decision in the design of the decoder is whether it shall find the solution in a single pass or use multiple passes over the search space to extract the best path. In the first pass of a multi-pass decoder relatively simple acoustic and language models can be used to find the part of the search space that is likely to contain the optimal path. This part of the search space is represented by a word lattice (see the following section for details). In the following passes more detailed and computationally more expensive models are used to find the final solution in the reduced search space. Another degree of freedom in the decoder design is the way in which the search space is traversed. The most common choices are time-synchronous and best-first search. For an extended discussion of possible implementation of a Viterbi MAP decoder see [Odell, 1995].

2.5 Word Lattices

Word Lattices are used as a compact representation of a large number of competing hypotheses of the spoken word sequence. As such they are ideally suited to store intermediate results in a multi-pass decoding approach. By constraining the space of permissible word sequences that are considered in the search, lattices can reduce the computational costs of further processing steps drastically. On the other hand they also limit the following decoding passes to those word hypotheses that are contained in the lattice. Errors made in the initial lattice generation can therefore (in general) not be recovered later on.

In addition to the use within a speech recognition system, lattices have also been used at the interface between the speech recognition component and the syntactic/semantic component in complex Natural Language Processing systems.

In the following word lattices are defined formally and the notation used in the rest of this thesis is introduced.

A word lattices is a labelled directed acyclic graph and is defined by:

\mathcal{E} the set of edges (links) in the graph

\mathcal{N} the set of nodes in the graph

$w : \mathcal{E} \rightarrow \mathcal{V}$ the labelling function which assign an unique word from the vocabulary to each edge

$t : \mathcal{N} \rightarrow \mathbf{R}^+$ assigns points in time to each node

$s : \mathcal{E} \rightarrow \mathcal{N}$ and $f : \mathcal{E} \rightarrow \mathcal{N}$ giving the start and final nodes of an edge

$p : \mathcal{E} \rightarrow \mathbf{R}$ assigns a probability/likelihood to each edge

The likelihood associated with each link is often further decomposed into components calculated from the acoustic model $p_{ac}(e)$ and the language model $p_{lm}(e)$.

If advanced acoustic or language models (such as triphones or a trigram model) are used then it has to be guaranteed that for each link in the lattice the context is unique to the extent required by these models (e.g. previous $n - 1$ words and neighbouring phones respectively). When applying more complex language models (for example applying a 4-gram to a trigram lattice) it becomes necessary to rebuild the lattice expanding the context where appropriate. The result is a drastic duplication of links corresponding to the same word and the same start and end times but slightly different acoustic or language model likelihoods due to the different contexts.

The advantage of this representation is that many of the lattice search and rescoreing problems can be solved by more general graph search procedures without having to explicitly refer to the original probabilistic models used to generate the lattice. This

approach is slightly different from the one presented in [Ortmanns et al., 1997, Wessel et al., 1998] where word graphs do not contain the language model scores, thus yielding more compact lattices but less straightforward rescoring algorithms.

2.6 Evaluation procedure

To assess the performance of a speech recognition system a suitable evaluation metric has to be defined. This metric is also used in the development and optimisation process of the system.

The evaluation is performed on a special set of utterances that have not been used in the training of the acoustic and language models. Such *unseen data* is used because the system normally performs much better on data that has also been used in the training process thus the estimate of the performance would be overly optimistic.

A simple evaluation metric is the *Sentence Error Rate* (SER, also called *string error rate*) which is calculated by comparing the hypothesis string generated by the decoder to the reference string and scoring the whole sentence as wrong if they differ. This metric is rather coarse as any deviation of the hypothesis from the reference is counted as one error independent of how similar the two strings actually are.

The by far most commonly used metric is the *Word Error Rate* (WER), which, as the name implies, measures the number of words that differ between the hypothesis and the reference. Alas the calculation of this metric is not as trivial as for the SER because the length of the hypothesis may differ from the reference. Therefore the WER is defined relative to an alignment of the two word sequences. In the alignment four different possible situations are distinguished:

correct The reference and hypothesis words are the same (or equivalent according to some set of rules taking e.g. spelling variants into account)

substitution The reference word is aligned to a different hypothesis word.

insertion An extra word has been inserted in the hypothesis that can not be aligned to a word in the reference.

deletion A word from the reference is missing completely.

The optimal alignment is found by a dynamic programming procedure that minimises the Levenshtein distance (the weighted sum of insertion, deletion and substitution errors) of the two word strings. The word error rate is then given as the number of errors divided by the number of reference words. Due to an excessive number of insertion errors the error rate can actually exceed 100%.

For most tasks the word based error metric is much more appropriate than the sentence based one. For example in a dictation system the subjective quality of the

recognition depends on the number of words the user has to correct manually and not the number of sentences in which the errors are located. Still there are applications where the string error rate is the more relevant metric. An example of this is the recognition of digit strings (typically telephone numbers or PINs). Here the misrecognition of even one digit can be catastrophic and render the whole hypothesis string useless. The only way to recover from an error is to repeat the whole string (this is typical in systems where speech is the only input modality, such as voice-dialing applications).

It should be noted that even the word error rate is very coarse, as all word errors are treated as equally important. In many domains there are errors which are much more grave than others. For example in a machine translation system missing one article in a sentence is presumably relatively unimportant whereas misrecognising the main verb will probably be fatal for the whole translation.

The metrics described above are only applicable to 1-best recognition hypotheses. The evaluation of multiple hypotheses stored as N-best lists or lattices is more complicated as an increase in the number of alternatives will usually improve the results. Therefore this number of alternatives has to be taken into account. One evaluation approach is to find the hypothesis with the smallest error rate among the alternatives and takes its error rate as the measure of the quality of the lattice/N-best list of a given size. The resulting metric is called *oracle error rate* as it is a lower bound on the error that can only be achieved if one invariably picks the optimum alternative (hence the name oracle). As a size metric of a lattice the number of links and nodes per word in the reference can be used. To illustrate the performance of a recogniser the oracle word error rate is plotted against the link density of the lattices produced by the decoder run with different pruning thresholds.

More details about the evaluation of the quality of a word lattice can be found in [Amtrup et al., 1997] where the disadvantages of different evaluation metrics are discussed and an alternative application-specific lattice size metric (appropriate for a NLP system) is presented.

3 Posterior Probabilities

In this chapter the estimation and use of posterior probabilities in the context of continuous speech recognition is discussed. The focus of this discussion is on using posterior probabilities of *words* instead of whole sentences. First the discrepancy between the word and sentence based metrics used in speech recognition is examined as a motivation to study the use of word based posteriors in the decoding process. In section 3.2 the concept of word level posterior probabilities is introduced informally and it is argued that in the tasks of *keyword spotting* and *confidence scoring* approximations of such posteriors can be employed. This is followed by a discussion of possible definitions of a word level posterior probability.

A way to actually estimate posterior probabilities for units below the sentence level from the output of a conventional decoder (i.e. a word lattice) is described in section 3.3. Finally an approach to combine these score to generate word level posteriors that can be used in a decoder is presented in section 3.4.

3.1 Sentence vs. Word Based Metrics

As mentioned in section 2.1 for the general pattern recognition problem the MAP criterion yields the optimal (Bayes) classifier in the sense that the probability of an error (wrong classification of a pattern) is minimised. This is obvious if the real posterior distributions $p(\omega_i|\mathbf{x})$ are used in the classifier (see [Fukunaga, 1990]). For most practical problems these real distributions are not known and have to be estimated from a training set. In [Nádas, 1985] it is shown that for the case of isolated word recognition the MAP criterion is still optimal if the real posteriors are replaced by appropriately estimated distributions.

The situation is different if the classification problem is to recognise a sequence of words. For such a task the MAP criterion only guarantees a decoder which is theoretically optimal with respect to the probability of assigning an incorrect label to the whole utterance (i.e. the sentence error rate). For many applications this is not an appropriate error metric. As mentioned above in a dictation application the number of words that are recognised incorrectly is far more relevant than the number of sentences that contain errors (e.g. two sentences with one incorrectly recognised word each are much easier to correct than one correct sentence and one with 5 incorrect words). For this

reason large vocabulary speech recognition systems are usually assessed by their *word* error rate instead of the *sentence* error rate.

Obviously the word and sentence error rates are correlated to some extent but this correlation is rather weak especially if the overall performance of the system is poor. In this case the sentence error rate is usually close to 100% anyway and changes in the word error rate have hardly any influence on the sentence error rate. If on the other hand the system's performance is relatively good and most sentences contain either no or only one or two errors, then the correlation is quite strong as additional word errors tend to produce new sentence errors as well.

Another important factor is the length (in words) of an average sentence. In the trivial case of one word sentences (effectively isolated word recognition) the word and sentence error rate are obviously equivalent. If a corpus contains more longer sentences then this correlation tends to become weaker.

To illustrate these points with concrete data, statistics from two continuous speech corpora (in the following named hub4 and hub5) were collected (see chapter 5 for a description of these corpora and the recognition system used).

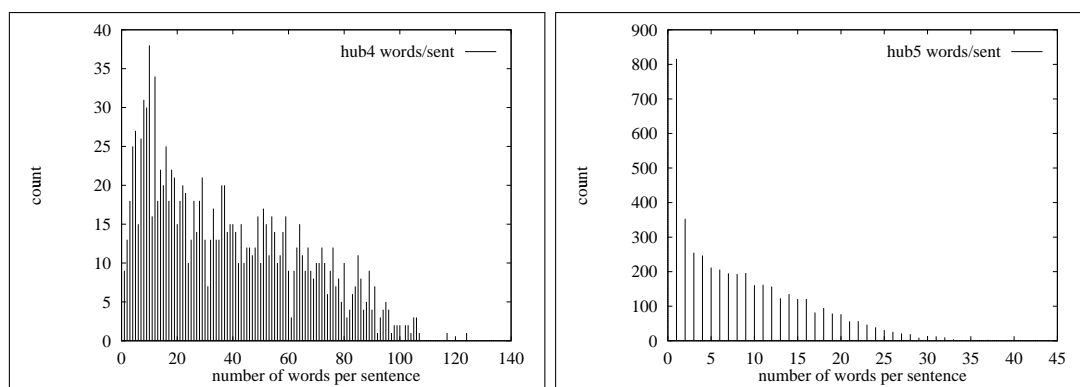


Figure 3.1: histogram of sentence length in the hub4 and hub5 corpora

In figure 3.1 histograms of the sentence lengths in words are shown for the two corpora.¹ It can be clearly seen that there are significant differences between the corpora. For hub4 (which consists of recordings of business news broadcasts) the average length of a sentence is 38 words, while for hub5 (spontaneous conversations over the telephone) the mean is only 9. In addition the distributions are very different, as for the telephone conversations a major part of the utterances consists of only one word (typically *yes*, *right* or *mhm*). The distribution for the news recordings has a peak at around 10 words.

¹It should be noted that here (and in the rest of the thesis) *sentence* refers to the segments of the audio signal that are fed as a unit to the decoder. Depending on the corpus these segments typically also contain (ungrammatical) fragments or multiple (linguistic) sentences.

From this it is to be expected that due to the shorter utterances in hub5 the correlation between the word and sentence error rates is much stronger.

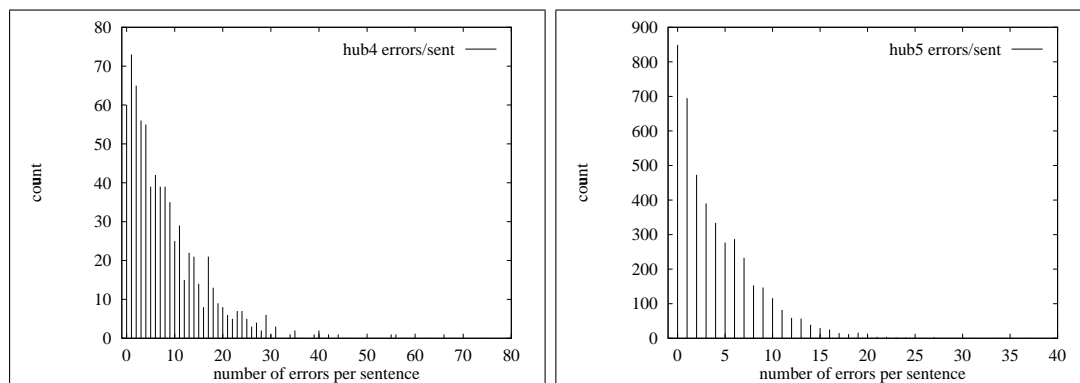


Figure 3.2: distribution of the number of word errors per sentence in the hub4 and hub5 corpora

The respective baseline recognition systems were run for both corpora and histograms of the number of word errors per sentence were produced (see figure 3.2). The shapes of the distributions look very similar for both sets, but it should be noted that the average number of incorrectly recognised words per sentence on the news recordings is 8.4 while it is only 4.1 on the telephone conversations despite the much higher overall word error rate on the latter corpus (see table 3.3).

	hub4	hub5
WER	17.4	43.0
SER	92.0	80.3

Figure 3.3: overall recognition performance for hub4 and hub5

The two test sets were randomly divided into small groups of sentences. For each such group the sentence and word error rate was measured to get a feel for the correlation between the two. The results are plotted in figure 3.4 (alas the number of sentences available for the hub4 set was much smaller than for hub5). While hardly very conclusive, these plots show that there is some correlation between the sentence and word error rates. But as expected this correlation is far from perfect and therefore minimising the number of sentence errors will not necessarily be optimal with respect to minimising the word error rate

The discrepancy between the use of a sentence based measure (the posterior $p(\mathbf{W}|\mathbf{X})$) in the decoder and the word based metric (WER) in the hypothesis evaluation suggests

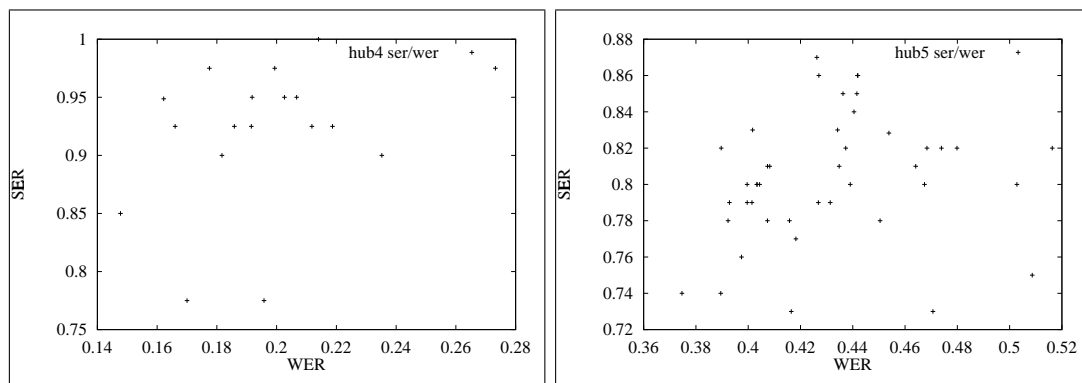


Figure 3.4: correlation of SER and WER

that using a more appropriate (word based) objective function in the decoder might improve the performance of the recogniser.

3.2 Word-level Posterior Probabilities

The above discussion suggests that it might be useful to investigate a measure that is more directly linked to the word error rate than the sentence posterior probabilities. The most obvious candidate would be a posterior probability on the level of individual words that directly indicates how likely it is that a particular words has been observed in an longer sentence.

In this section two obvious applications related to speech recognition that can make use of such a word level posterior probability are discussed to informally introduce the idea.

3.2.1 Keyword Spotting

The use of a word level posterior in the context of large vocabulary speech recognition was first suggested in [Weintraub, 1995] under the name of “LVCSR log-likelihood ratio scores”. Weintraub describes the use of these scores in the framework of a keyword spotting task where the aim is to reliably detect the presence of certain keywords in long streams of audio recordings or in many smaller separate utterances.

Weintraub demonstrated that a well trained large vocabulary speech recognition system (the specific system had a vocabulary of 5,000 entries) can be successfully employed to improve keyword spotting performance compared to traditional systems.

Any normal speech recogniser can be adapted for this keyword spotting algorithm if the recogniser vocabulary contains all the desired keywords. The log-likelihood score is

defined as:

$$l_k = \frac{\sum_{\mathbf{W} \in \mathcal{W}_k} p(\mathbf{W}|\mathbf{X})}{\sum_{\mathbf{W}} p(\mathbf{W}|\mathbf{X})} \quad (3.1)$$

where \mathcal{W}_k is the set of all word sequences that contain the keyword k . Thus the sum in the numerator is the probability of all sentence level posteriors for sentences containing the relevant keyword while the denominator is used to normalise this by the probability of all possible word sequences. The sentence level posteriors are calculated in the usual way as:

$$p(\mathbf{W}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{W})P(\mathbf{W})}{p(\mathbf{X})} \quad (3.2)$$

where the probabilities in the numerator are given by the acoustic and language model respectively. Inserting this into equation 3.1 yields (cancelling the probability of the observation sequence $p(\mathbf{X})$):

$$l_k = \frac{\sum_{\mathbf{W} \in \mathcal{W}_k} p(\mathbf{X}|\mathbf{W})P(\mathbf{W})}{\sum_{\mathbf{W}} p(\mathbf{X}|\mathbf{W})P(\mathbf{W})} \quad (3.3)$$

In [Weintraub, 1995] the sums in equation 3.3 are further simplified by summing only over those word sequences which are found in an N-best list generated by the Viterbi decoder. Only this simplification makes the computation of the sums feasible but depending on the number of alternative hypotheses considered and the accuracy of the recogniser this might introduce extra errors.

A major complication in the evaluation of the likelihood-ratio scores is the fact that one keyword can appear at different (time) position in different N-best entries. To resolve this problem the N-best word sequences were augmented with the time information of the corresponding Viterbi segmentation. Two occurrences of the same keyword are treated as separate keyword hypotheses if they do not overlap in time. In such cases a separate likelihood-ratio score is calculated for each keyword hypothesis.

One advantage of this score over other keyword spotting approaches is that it takes the prior probabilities encoded in the language model into account. This is a significant improvement over conventional keyword spotting system which commonly only used acoustic information. Typically those systems use two different acoustic models: one that models the characteristics of the keyword and one so-called background model that is used for normalisation purposes. In Weintraub's approach no explicit background model is necessary but instead all the word/phone models of the large vocabulary speech recognition system are used. If a sufficiently large vocabulary and large number of alternative N-best hypotheses is used, one might expect the normalisation (the denominator

in equation 3.3) to work at least as reliably as the explicit background models in the conventional approach.

The LVCSR technique tends to perform better than the simple two model approach but depending on the application the increased overhead of training and running an LVCSR system might be unacceptable.

3.2.2 Confidence Scores

A related task where word posterior probabilities are more directly applied is the estimation of confidence scores.

As the output of a speech recogniser usually contains errors it can be interesting for many applications to have additional information how “confident” the recogniser is in the accuracy of the recognition result. This is typically encoded as a confidence score in the range between 0.0 and 1.0. This score could be assigned at different levels (e.g. utterance, word or phone level). Most commonly such scores are assigned at the word level to the recognition results. If these scores accurately reflect the probability that a word has been recognised correctly then they can be used in a wide variety of applications. For example they could be used to reject parts of an utterances in a dialogue system that were poorly recognised or detect out-of-vocabulary words. They could also be used to decide which portions of an utterance should not be used in an unsupervised acoustic adaptation scheme (e.g. MAP or MLLR).

Confidence scores are closely related to the problem of keyword spotting, as a robust confidence estimator could also be utilised to identify the occurrence of the keywords.

A typical confidence scoring system consists of two separate stages (see e.g. [Cox and Rose, 1996]). The first stage is the extraction of suitable features from the decoding process or directly from the acoustic signal. These features are then combined to yield the confidence score. For this combination various techniques have been successfully used. Among them are simple Bayesian models, generalised linear models, decision trees and neural networks.

A procedure to evaluate the quality of a confidence estimator was introduced in [Siu et al., 1997]. The hypothesised word sequence is aligned with the reference transcription (e.g. using the Levenshtein distance mentioned in section 2.6). Based on this alignment each hypothesised word w_i is labeled as either correct ($c_i = 1$) or incorrect ($c_i = 0$). In [Weintraub et al., 1997] a minimum overlap of 50% between the hypothesis and the reference word is used as an additional constraint before labelling words as correct. The quality of a set of confidence scores q_i is now captured in the following information theoretic measure (normalised cross entropy, see [Siu et al., 1997] for details):

$$NCE = \frac{H(C) - H(C|X)}{H(C)} \quad (3.4)$$

where $H(C)$ is the entropy of the sequence of tags c_i and $H(C|X)$ is the entropy of the confidence scores q_i (or rather in the information theoretic view the conditional entropy of the confidence tags given the features used). Essentially the numerator measures the additional information (in bits) provided by the confidence scores compared to the baseline case of setting the confidence scores to the *a-priori* probability of correctness (sum of substitutions and insertions divided by the number of recognised words). This is normalised by the entropy of the priors to compensate for effects of the overall word error rate of the recogniser.

Features typically used for the estimation of confidence scores include the (normalised) likelihoods derived from the acoustic and language models and word dependent measures such as the length in phones. Another feature that is often successfully used is known under the name of “N-best homogeneity” and is actually very similar to the likelihood-ratio discussed above in the context of keyword spotting (equation 3.3). The only difference is that the decision which word instances are considered as equivalent is not based on the time overlap but on a Levenshtein alignment of each N-best entry with the recognition output (i.e. the 1-best entry).

It has been observed that the N-best homogeneity alone performs remarkably well and that adding other features to the estimator does not improve the normalised cross entropy much further.

3.2.3 Definition

The sentence posterior probability is straightforwardly defined as the probability of the word sequence given the acoustic feature vectors: $p(w_1 \dots w_n | \mathbf{X})$. By definition the sentence hypothesis covers the whole series of feature vectors.

The situation for a word level posterior probability is more complicated as the boundaries of the word in question are not a-priori known. Depending on the application different variants of a word posterior might be useful. The following lists some of these variants and discusses their differences.

The simplest way to define a word posterior is to treat the start and end time as additional random variables. This leads to posteriors of the form: $p(w|t_s, t_e, \mathbf{X}_{t_s}^{t_e})$. The calculation of these values can be achieved relatively easily (see section 3.3) but as for many applications the concrete start and end times are not relevant, it is desirable to find a more general definition that does not involve the specific times.

An alternative is to consider the posterior probability, that a word spans a current point in time. This could be easily calculated from the previous suggestion:

$$p(w|t, \mathbf{X}) = \sum_{t_s \leq t \leq t_e} p(w|t_s, t_e, \mathbf{X}) \quad (3.5)$$

Possible applications of this time dependent word posterior will be discussed in section 3.4

If the time information is not relevant at all then it is useful to define the word posterior probabilities relative to a particular sentence hypothesis. This way it is possible to define the posterior probability as the probability that a word in the hypothesis is scored as correct when aligned to the word sequence actually spoken (i.e. the reference transcription). This is the kind of posterior typically assumed for the purpose of confidence scoring. Following [Siu et al., 1997] this could be defined as $p(w_i, c_i = 1 | \mathbf{X}, \mathbf{W})$, where c_i is the tag assigned to word w_i by the scoring procedure (i.e. DP alignment with Levenshtein metric and/or minimum time overlap between hypothesis and reference as suggested in [Weintraub et al., 1997]).

3.3 Calculating Posterior Probabilities in Lattices

In this section an approach to estimating posterior probabilities in a speech recogniser will be discussed. To make this estimation feasible it will be based on the information present in a lattice produced by a conventional Viterbi MAP decoder.

In a word lattice each link is labelled with a word, the start and end times, and the likelihood score calculated from the acoustic and language models:

$$p(w_i, \mathbf{X}_{t_s}^{t_e} | w_1 \dots w_{i-1}, w_{i+1}) = p_{ac}(\mathbf{X}_{t_s}^{t_e} | w_i, w_{i-1}, w_{i+1})^{\frac{1}{\gamma}} p_{lm}(w_i | w_1 \dots w_{i-1}) \rho^{\frac{1}{\gamma}} \quad (3.6)$$

The likelihood of a link is actually the joint probability of the word and the acoustic observations in the time interval (t_s, t_e) given all previous words and the following word. The dependence on all the previous words is necessary as these form the “history” for the language model probability. Because the acoustic model might incorporate cross-word models (e.g. triphones) the following word has to be added as well.

To be able to have only one likelihood at each link, the lattice must have such a structure that all the variables in the probability are defined uniquely. This is guaranteed by generating separate nodes for each time depending on the word history. This would actually lead to a tree structured lattice but when a N-gram language model is used, paths can recombine if they both end in the same $n - 1$ word history.

It should be noted that in equation 3.6 the scaling of the acoustic and language model is performed differently from the way that is customary in a Viterbi decoder. Instead of scaling the language model down as usual, here the acoustic score is scaled up and the word insertion penalty is adjusted accordingly. This is done because in the following calculations sums of these scores will be calculated instead of just picking the best score as in a Viterbi decoder. If these scores are added it is better to adjust the dynamic range of the acoustic model to that of the language model which after all

estimates “real” probabilities instead of probability density values and also does not suffer as severely from the underestimation problem mentioned in section 2.4.

From these link likelihood scores posterior probabilities $p(l|\mathbf{X}_{t_s}^{t_e})$ can be calculated for each link l . This is achieved by an algorithm very similar to the forward-backward algorithm used to train HMMs. For each node in the lattice two quantities are defined: the forward probability $\alpha(n)$ and the backward probability $\beta(n)$.

The forward probability $\alpha(n)$ represents the sum of the likelihoods of all paths from the start node of the lattice to node n . This is not equivalent to the sum over all word sequences preceding the node in question due to the fixed unique history constraint described above. Analogously $\beta(n)$ is the sum over all paths to the end node.

More formally the forward probability can be defined recursively as the sum over the likelihoods of all links ending in n multiplied by the forward probabilities of their start nodes:

$$\alpha(n) = \sum_{l, e(l)=n} p(l)\alpha(s(l)) \quad (3.7)$$

This can be efficiently calculated by traversing the nodes in topological order (i.e. never visiting the end node of a link before the corresponding start node) and propagating the probabilities multiplied by the link likelihood forward along all outgoing links. At each node the probabilities propagated along the incoming links are combined by addition. The backward calculation can be performed by an analogous backwards traversal of the lattices.

From these forward and backward probabilities the posterior probability of a link can be defined:

$$p(l|\mathbf{X}) = \frac{\alpha(s(l))p(l)\beta(e(l))}{p(\mathbf{X})} \quad (3.8)$$

The probability $p(\mathbf{X})$ can be calculated as the sum of all the sentence posterior probabilities:

$$p(\mathbf{X}) = \sum_{\mathbf{W}} p(\mathbf{W}, \mathbf{X}) \quad (3.9)$$

As it is infeasible to actually sum over all possible word sequences, this can be approximated by only considering the subset of paths that are contained in the lattice. This sum over all paths is obviously equal to the forward probability of the end node $\alpha(n_e)$ and the backward probability of the start node $\beta(n_s)$.

This approximation is relatively good as the sum over all paths tends to be dominated by a small proportion of all possible paths. Figure 3.5 shows a graph of the posterior

probability against the size of a lattice (number of paths). This graph was produced by successively pruning a lattice from the hub4 corpus and at each stage counting the number of paths and calculating the forward probability of the final node. Both axes in the graph are logarithmic, on the x-axis base 10 logarithms of the number of paths through the lattice are used.² On the y-axis the natural logarithm of the probability estimated according to equation 3.9 is plotted. It should be noted that the utterance from which the lattice was produced is one of the longest in the testset (ca. 30 seconds), which explains the large number of paths (almost $4 \cdot 10^{66}$). Despite the extremely large numbers it can be clearly seen that the estimate of the posterior probabilities converges. Therefore the approximation to restrict the summation in 3.9 to the high scoring paths seems acceptable (e.g. the difference between the posterior estimated from 10^{45} paths does not differ much from the one estimated based on 10^{66} paths).

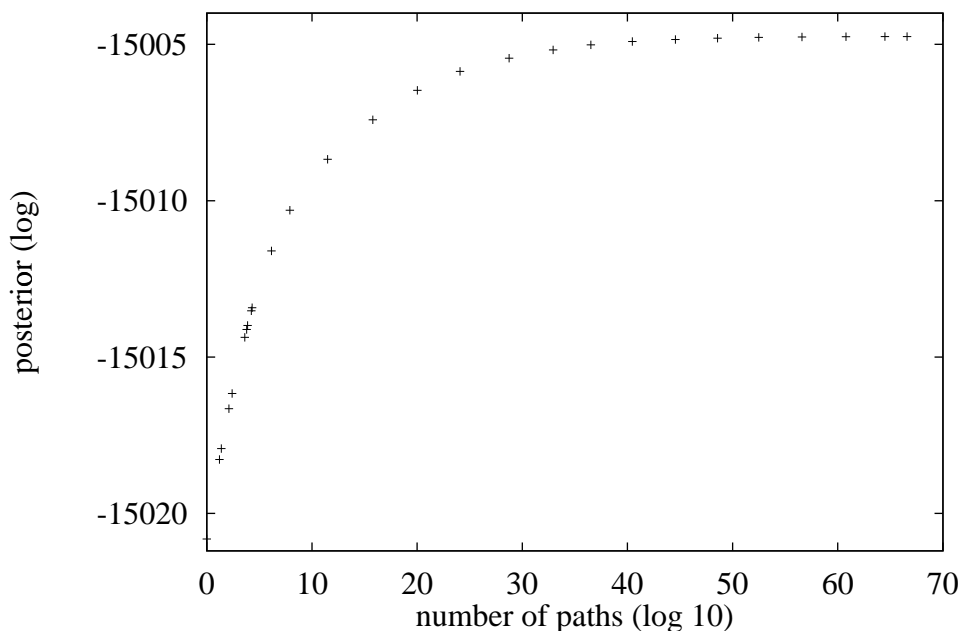


Figure 3.5: posterior probability estimate against lattice size

The probabilities estimated according to equation 3.8 can for example be directly used to prune lattices down to smaller sizes while still retaining those parts of the lattices that contain the “most probable” paths. This is an alternative to the normal best-path

²Here it is important to realise that the number of paths through a lattice is approximately exponential in the number of links and nodes (which determines the size of the lattice when stored in memory or on disk).

pruning which only takes the highest scoring path through each link into account. See section 5.2 for results based on an implementation of both pruning methods.

3.4 Time Dependent Word Posteriors

The link posterior probabilities described in the previous section are only of limited direct use, as there usually are a large number of such links for every word hypothesis at every point in time (due to the N-gram expansion explained above). Therefore these link probabilities need to be combined to word probabilities. As explained in section 3.2.3 the easiest way to do this is by taking the start and end times of the hypothesis as extra variables and adding the link probabilities:

$$p(w|t_s, t_e, \mathbf{X}) = \sum_{\{l|s(l)=t_s, e(l)=t_e, w(l)=w\}} p(l|\mathbf{X}) \quad (3.10)$$

The problem with these posteriors is that they are too specific, because they depend on the exact boundary times. A solution to this is to calculate posterior probabilities for each point in time in the utterance. Informally speaking these “instantaneous” word posterior distributions capture which words the decoder considers likely at that particular time.

The big advantage of this approach over a normal Viterbi score is that it does not only take the best path into account but instead also encodes information about the number and likelihoods of all alternatives (different segmentations of the same word and different words). For these reasons these distributions can be expected to provide additional valuable information for rescoreing the hypotheses of a Viterbi MAP decoder.

The calculation from the link posteriors is simple:

$$p(w|t, \mathbf{X}) = \sum_{\{l|s(l) \leq t \leq e(l), w(l)=w\}} p(l|\mathbf{X}) \quad (3.11)$$

Here it should be noted that this summation over all link probabilities spanning a particular time guarantees that $p(w|t, \mathbf{X})$ is actually a probability distribution, i.e. $\sum_w p(w|t, \mathbf{X}) = 1$.

This can be easily shown by considering the lattice in figure 3.6. As explained in the previous section the link posteriors $p(l|\mathbf{X})$ have been normalised by the utterance probability $p(\mathbf{X})$, which has been calculated as the sum over all paths through the lattice. By summing over all links spanning a specific time all paths through the lattice are included in the sum:

$$\sum_{\{l|s(l)\leq t\leq e(l)\}} p(l|\mathbf{X}) = 1 \quad (3.12)$$

As each link is labelled with exactly one of the words in the vocabulary it follows that $\sum_{w\in\mathcal{V}} p(w|t\mathbf{X}) = 1$

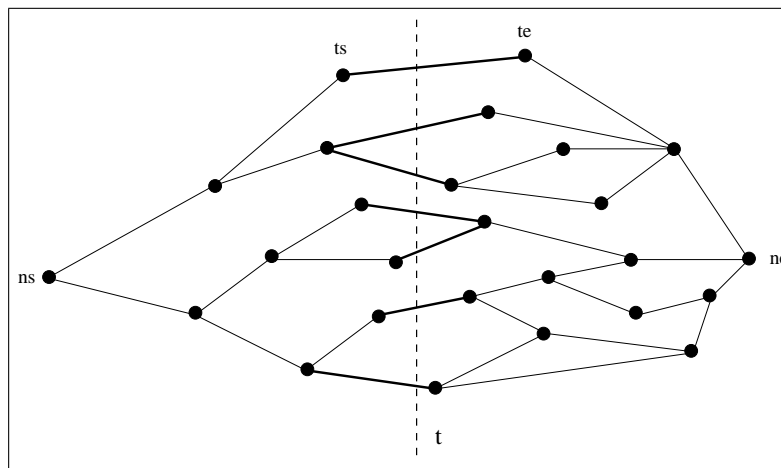


Figure 3.6: time dependent posterior

The time dependent word posteriors can be used as an extra likelihood factor in the decoder. The conventional Viterbi score (see equation 2.13) is augmented by the product of the word posteriors of the words in the path:

$$f(\mathbf{W}) = p(\mathbf{X}, \hat{\mathbf{q}}|\mathbf{W})^{\frac{1}{\gamma}} P(\mathbf{W}) \rho^{\frac{\|\mathbf{W}\|}{\gamma}} \prod_{t=0}^T p(w(\hat{\mathbf{q}}, t), t|\mathbf{X}) \quad (3.13)$$

where $w(\hat{\mathbf{q}}, t)$ is the word corresponding to time t in the path $\hat{\mathbf{q}}$.

Experimental results of this rescoring technique will be presented in chapter 5.

The time dependent posterior probabilities can be seen as a generalisation of the set of techniques that record the number of different hypotheses a 1-best MAP decoder keeps active in each frame. Due to the beam pruning employed in the decoder this number of active models varies significantly over the utterance. It has been observed that especially at the beginning of words the number of active hypotheses with similar scores is very high as the decoder is still “uncertain” about the identity of the word and has to consider many alternative continuations. This number tends to decrease significantly towards the end of words (see [Odell, 1995] for a discussion on how to

exploit this for an improved pruning scheme). This measure has also been successfully employed as a feature for confidence scoring (e.g. [Schaaf and Kemp, 1997]) and for the detection of out-of-vocabulary words (a comprehensive discussion of this can be found in [Fetter, 1997]).

The advantage of the time dependent word posteriors over the simple count of active models is that it provides a *weighted* measure of the “certainty” of the decoder, i.e. the actual likelihood values are utilised instead of just counting the number of hypotheses above the pruning likelihood threshold. Another important advantage is that the word posteriors incorporate information about the continuations of a hypothesis from the current frame to the end of the utterance. The number of active models in a time-synchronous MAP decoder can by definition only take the previous history of frames into account. This is the difference between using just forward scores (as in the time-synchronous decoder) and calculating estimates of the forward-backward probabilities in a rescoring paradigm.

It should be noted that in [Wessel et al., 1998] similarly defined word posterior scores are used as confidence scores of the words in the 1-best hypothesis string of a conventional MAP decoder. Here for each link that is part of the MAP path the maximum of the time dependent posteriors of the corresponding word is taken as the confidence score for that word. The approach described in this section was developed independently and can be seen as an extension in so far as the scores of all frames are combined and (more importantly) used in a rescoring paradigm to actually define a new objective function that leads to improved decoding results.

4 Minimising Word Error Rate

As explained in section 3.1 using the MAP criterion in the decoder only results in achieving the minimum sentence error rate. In this chapter an approach to deriving an optimal decoding function from the desired error metric (WER) will be described.

The general approach (first presented in [Stolcke et al., 1997]) is explained in section 4.1, followed by the discussion of two alternative ways of implementing it in sections 4.2 and 4.3 respectively.

4.1 Expected Word Error Rate

Given the fact that the evaluation criterion of our speech recogniser is the number of word errors, it seems sensible to develop a decoder that uses this metric directly to find the optimal word sequence. In the following the technique introduced in [Stolcke et al., 1997] will be described.

The optimal decoder is the one that for each potential utterance (i.e. sequence of acoustic observations \mathbf{X}_1^T) always returns the word sequence \mathbf{W} with the lowest expected word error rate:

$$E[WER(\mathbf{W})|\mathbf{X}] = \sum_{\mathbf{W}_{ref}} WER(\mathbf{W}, \mathbf{W}_{ref})P(\mathbf{W}_{ref}|\mathbf{X}) \quad (4.1)$$

where $WER(\mathbf{A}, \mathbf{B})$ is the word error rate of \mathbf{A} if compared to the reference \mathbf{B} (in general $WER(\mathbf{A}, \mathbf{B}) \neq WER(\mathbf{B}, \mathbf{A})$ as the underlying Levenshtein distance can be asymmetric).

It is important to note that the formulation in equation 4.1 is **not** in terms of a set of acoustic observations with associated reference transcriptions. As mentioned in section 2.1 there is no deterministic mapping from a given acoustic observation \mathbf{X} to the word sequence. One observation sequence can correspond to different word sequences. Therefore we have to express the expected word error rate in terms the posterior probability $P(\mathbf{W}_{ref}|\mathbf{X})$ ¹. At this point the posterior used is the true posterior (which is obviously unknown) and not some estimate from a model.

¹This is a useful formulation even if the mapping were deterministic (i.e. $P(\mathbf{W}_{ref}(\mathbf{X})|\mathbf{X}) = 1$) as we do not actually know the reference word sequence in this optimisation.

The optimal decoder g_{opt} is now trivial to state:

$$g_{opt}(\mathbf{X}) = \underset{\mathbf{W}}{\operatorname{argmin}} E[WER(\mathbf{W})|\mathbf{X}] \quad (4.2)$$

In this formulation the word error rate could be replaced by any appropriate metric. For example in [Goel and Byrne, 1999] several metrics based on the performance in an Information Extraction task have been suggested. Instead of just measuring the Levenshtein distance of the two strings, a more complex distance function is evaluated. This might involve some fairly (computationally) expensive processing if for example the complex information extraction backend needs to be run on the two word sequences. This way the actual performance of the whole system can be used as an optimisation criterion but this approach may be prohibitive for computational reasons. Therefore it is desirable to find a simple metric that can be evaluated efficiently but still is highly correlated to the actual desired task specific metric used to assess overall system performance. The examination of such metrics is beyond the scope of this work and in the following only the word error rate will be considered.

It is infeasible to directly use the new optimisation criterion in the form given above as it involves a minimisation (in equation 4.2) and a summation (in equation 4.1) over all possible word sequences. Two alternative approaches to restrict the set of word sequences to be considered will be discussed in the two following sections. Another problem is that the calculation of the expected word error rate involves a weighted average of word error rates, where the weights are the (sentence) posterior probabilities. As these are unknown they must be replaced by appropriate estimates based on the models available.

4.2 N-best List based Minimisation

In [Stolcke et al., 1997] an approach to approximating the optimal decision criterion is presented which is based on N-best lists.

The idea is to train a speech recogniser in the normal Maximum Likelihood way and then run a conventional Viterbi MAP decoder on the unknown utterance to produce a list of the N-best scoring hypotheses. This N-best list is used as the basis for implementing the optimal decoding approach. The new decoding objective function is only evaluated for the word sequences contained in the N-best list. This rescored approach is similar to the multi-pass decoding described in 2.4 and suffers from the same problem, namely the fact that the choice of the decoder is limited to the word sequences contained in the N-best list.

To get an impression of how much of a limitation this actually is, a MAP decoder was run on the hub4 testset (see section 5.1) and N-best lists of different sizes were produced. For each of these N-best lists the oracle word error rate was determined. This error rate is an upper bound on the performance of a N-best rescored decoder. An additional

experiment was run in which an oracle was used that extracted the worst hypotheses (according to WER) from the list. This lower bound on the performance provides a measure of the amount of variation in the N-best hypotheses. The results of these oracle experiments are summarised in figure 4.1.²

N	oracle		anti-oracle
	SER	WER	WER
1	92.0	19.8	19.8
5	87.5	17.6	22.7
10	85.3	16.9	23.9
15	84.9	16.4	24.5
20	84.5	16.1	25.0
25	83.7	15.9	25.3
50	81.7	15.2	26.5
75	80.6	14.9	27.2
100	80.1	14.6	27.7
150	79.8	14.3	28.2
200	79.4	14.0	28.7

Figure 4.1: oracle WER on hub4 N-best lists

It can be clearly seen that even a moderately sized N-best list provides ample room for improvement (alas it is also possible to increase the error rate quite significantly). This is illustrated in figure 4.2 where the oracle word error rates (best and worst respectively) are plotted against the number of hypotheses in the N-best list. From this graph it is obvious that the usefulness of N-best list is limited, i.e. until about 100 alternatives the oracle WER falls rapidly but beyond that point the WER falls only very slowly in terms of N .

In the implementation based on an N-best list the optimal decoder takes the following form (see [Stolcke et al., 1997]):

$$\hat{g}_{opt}(\mathbf{X}) = \underset{\mathbf{W} \in \mathcal{W}_{n-best}}{\operatorname{argmin}} E[WER(\mathbf{W})|\mathbf{X}] \quad (4.3)$$

$$= \underset{\mathbf{W} \in \mathcal{W}_{n-best}}{\operatorname{argmin}} \sum_{i=1}^n WER(\mathbf{W}, \mathbf{W}_i) P(\mathbf{W}_i|\mathbf{X}) \quad (4.4)$$

where $\mathcal{W}_{n-best} = \mathbf{W}_1, \dots, \mathbf{W}_n$ is the set of the n best hypotheses from the MAP decoder.

²The word error rates given in this table were measured according to the HTK scoring convention as explained in section 5.1.

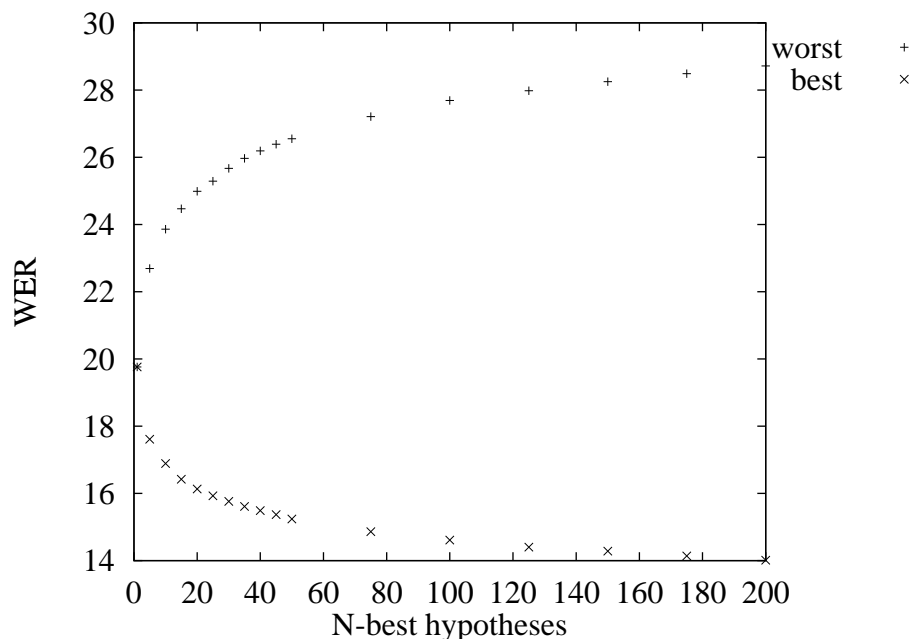


Figure 4.2: best and worst (oracle) WER in N-best lists for hub4

The minimisation and the weighted summation have been restricted to the N-best lists, which still leaves the problem of estimating the posteriors $P(\mathbf{W}_i|\mathbf{X})$. These can also be approximated by information from the N-best list, namely the scores produced by the MAP decoder:

$$P(\mathbf{W}_i|\mathbf{X}) = \frac{P(\mathbf{W}_i, \mathbf{X})}{\sum_{\mathbf{W}_i \in \mathcal{W}_{n-best}} P(\mathbf{W}_i, \mathbf{X})} \quad (4.5)$$

where the joint probability is calculated in the normal way by combining acoustic and language model scores. As discussed in section 3.3 in this case it is important to scale down the acoustic score to compensate for the effects of the independence assumption in the parameter estimation. In all experiments the grammar scale γ that had been previously optimised for the MAP decoder has been used:

$$P(\mathbf{W}_i, \mathbf{X}) = p_{ac}(\mathbf{X}, \hat{\mathbf{q}}|\mathbf{W}_i)^{\frac{1}{\gamma}} P_{lm}(\mathbf{W}_i) \rho^{\frac{\|\mathbf{W}_i\|}{\gamma}} \quad (4.6)$$

The optimisation in equation 4.3 is relative costly to perform since the computational complexity rises with $\mathcal{O}(N^2)$. This is due to the fact that in the minimisation an expected

WER is estimated for every one of the N alternatives. Each of these estimates involves a summation over N posteriors and N Levenshtein alignments.

To assess the impact of the size of the N -best list on the estimate of $E[WER(\mathbf{W})|\mathbf{X}]$ an experiment was run on a lattice from the hub4 corpus. N -best lists of different sizes were produced and the expected word error rate of the 1-best hypothesis was separately estimated based on each of these. The results are shown in figure 4.3. Each curve corresponds to an estimate based on one of the N -best lists, e.g. the leftmost curve is based on a 200-best list while the rightmost used 3000 alternative hypotheses. One curve represents the partial sums over the posteriors in equation 4.4 with increasing i . The fact that the leftmost curve ($N = 200$) rises much more sharply than the rightmost ($N = 3000$) is due to denominator in equation 4.5, which obviously increases when a larger N is used, which results in the horizontal scaling. The upper endpoints of the curves correspond to the estimate of $E[WER(\mathbf{W})|\mathbf{X}]$ for the respective values of N .

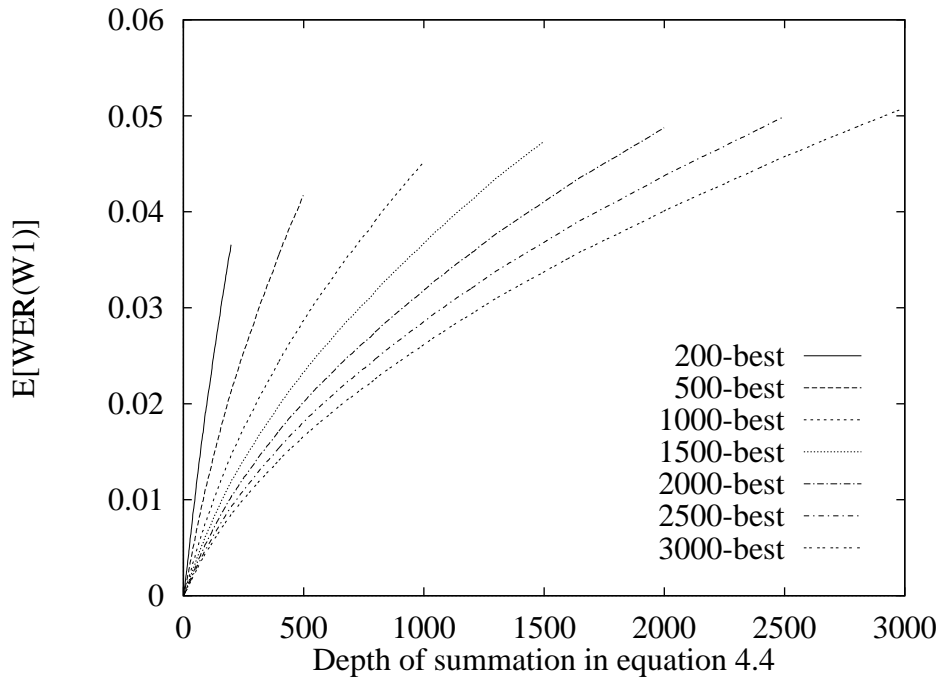


Figure 4.3: $E[WER]$ based on different N -best list

The following conclusions can be drawn from this graph:

- For a fixed N the partial sum in the estimate of $E[WER(\mathbf{W})|\mathbf{X}]$ rises rather sharply, indicating that the posteriors of the best hypotheses dominate the estimate

- The “heights” of the endpoints of the different curves (i.e. the actual estimates for a particular N) do not vary much. Even more importantly the difference in “height” seems to decrease with increasing N . Increasing the number hypotheses considered from 1000 to 3000 doesn’t seem to have a drastic influence on the estimate.
- The word error rate is underestimated quite severely. In this example the estimate is 5% while the actual error rate when compared to the reference is 40%. This discrepancy is not quite as severe for shorter utterances.

In all of the following experiments reported here only the top 1000 sentence hypotheses generated by the MAP decoder were used as the basis for the posterior distribution.

Another simplification to speed up the rescoring (suggested in [Stolcke et al., 1997]) is to only consider part of the N-best list as potential candidates to become the result of the rescoring. The idea is to only estimate $E[WER(\mathbf{W})|\mathbf{X}]$ for some of the top ranking alternatives while still performing the sum over the posterior distribution based on a larger N-best list. After this additional modification the decoding function becomes:

$$\hat{g}_{opt}(\mathbf{X}) = \underset{\mathbf{W} \in \mathcal{W}_{k-best}}{\operatorname{argmin}} \sum_{i=1}^n WER(\mathbf{W}, \mathbf{W}_i) P(\mathbf{W}_i | \mathbf{X}) \quad (4.7)$$

where $\mathcal{W}_{k-best} = \mathbf{W}_1, \dots, \mathbf{W}_k$ contains only the k top hypotheses with $k \ll N$. This reduces the computational complexity to $\mathcal{O}(kN)$. In chapter 5 experimental results based on this optimisation are presented.

In [Goel et al., 1998] it is suggested to add exponential weighting factors to the word error rate and the sentence posterior in equation 4.7. The idea is to tune these factors on a separate testset in the hope that this modification of the estimate will compensate for the suboptimal posteriors. Some experiments with these extra weights were performed and no consistent significant improvement could be achieved. Therefore the experiments described in section 5.3.1 use equation 4.7 directly without any additional weights.

4.3 Confusion Networks

It would be desirable to perform the minimisation described in the previous section directly on a word lattice instead of using an N-best list. Word lattices can compactly represent far more different hypotheses than N-best lists. In general the number of paths through a lattice is exponential in the number of links. These paths all corresponds to different segmentation hypotheses (i.e. word sequence plus boundary times) of the utterances. In a conventional N-best list only the best (i.e. highest scoring) segmentation of each word sequence is represented. In the rescoring paradigm discussed here this actually poses a problem as all the alternative segmentations of the word sequences

contained in the N-best list are ignored. Therefore the posterior probabilities of all word sequences will be severely underestimated and thus the estimate of the word error rate will be skewed in an unpredictable way. This problem could be avoided by considering all the (high scoring) competing segmentations of the best word sequences. This would either drastically increase the size of the N-best list (and hence incur high costs in extra computation) or if N is kept constant massively reduce the number of different word hypotheses considered. Therefore an approach that avoids the explicit expansion into N-best list but operates directly on the lattice is needed.

For a lattice rescoring algorithm to work effectively it is necessary to share computation by exploiting the structure of the lattice. Ideally an algorithm is able to reuse the calculation required for all paths that share a common prefix (or suffix). Thus it is possible to compute some measures only once for each node (or link) and then combine these as necessary. A prototypical example is the forward-backward algorithm used in section 3.3 to calculate link posteriors in a lattice.

The implementation of the decoding approach discussed in the previous section based on a lattice is difficult for the following reasons:

- The WER has to be calculated between two complete word strings and can (in general) not be decomposed into parts which can be shared between common prefixes, i.e. the alignment in the prefix can change based on the continuations.
- The estimate of the expected word error rate is a sum over the product of the WER and the posterior of the path. This sum can not be easily separated into parts from which a common term can be factored out. This is due to the fact that the product terms prevent factoring.

A method to overcome these problems has been suggested in [Mangu et al., 1999]. The word lattice is first transformed into a special form in which the calculation of the expected word error rate becomes trivial. This special form of a lattice is called a *confusion network* (an example is shown in figure 4.4). Each link is labelled with a word and a probability (omitted in the example).

The most important feature of these lattices is that they are linear, in the sense that every path from the start to the end node has to pass through all nodes. A consequence of this (combined with the acyclicity) is that all paths between two nodes have the same length. Thus the confusion network naturally defines an alignment for all pairs of paths (called a “multiple alignment” by Mangu). This alignment is used as the basis for the word error rate calculation. The “-” entries in the network correspond to deletions in the alignment. It must be noted that this multiple alignment is suboptimal as it can overestimate the word error rate of two paths compared to the exact pairwise alignment. In practice this is not a significant problem.

To estimate the word error rate, sets C_i are defined that contain all the links that connect the node n_i with the next node. Thus each C_i contains the alternative hypotheses

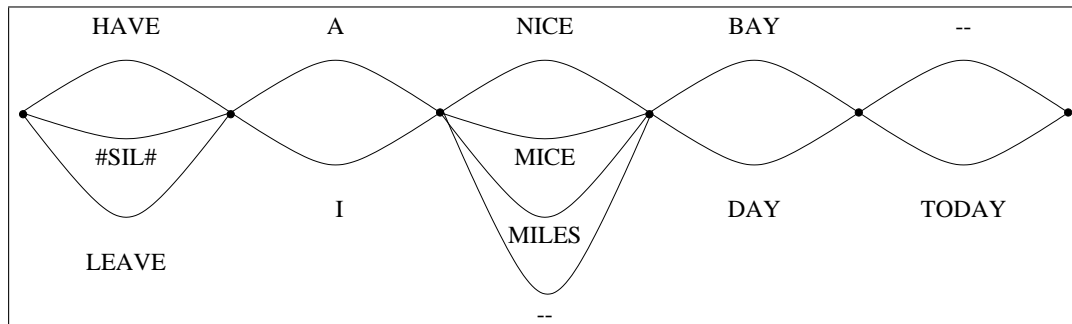


Figure 4.4: example confusion network

for the the word in position i in the final word string. The posteriors of the elements of each set add to one.

The expected word error rate of a hypothesis \mathbf{W} can be calculated as the sum of the expected error rates for each set C_i (based on [Mangu et al., 1999]):

$$E[WER(w_1, \dots, w_n) | \mathbf{X}, CN] = \sum_i^n \left(\sum_{l \in C_i, w(l) \neq w_i} p(l) \right) \quad (4.8)$$

Minimising this quantity is achieved trivially by picking the word with the highest posterior from each set.

An approach to actually construct a confusion network from a word lattice is also presented in [Mangu et al., 1999]. The task is treated as a clustering problem, where the links from the original lattice have to be clustered into the sets C_i mentioned above. To achieve a sensible clustering the algorithm is constrained to keep the precedence ordering in the original lattice intact. If a link l_a precedes a link l_b in the lattice (i.e. there is a path from l_a to l_b) then the cluster in which l_a ends up must also precede l_b 's cluster. Very informally speaking this ensures that the order of words stays the same, the lattice is just ‘‘collapsed’’ along the vertical axis.

More formally this can be ensured by defining a precedence relation on the sets of links. A set C_1 precedes C_2 if any of C_1 's members precede any of C_2 members in the original lattice. By combining clusters additional precedences are introduced. Starting from the partial order defined by the word lattice this is repeated until a total order of the clusters is reached, which corresponds to a linear structure of the lattice.

The algorithm implemented based on [Mangu et al., 1999] consists of the following steps:

1. calculation of all link posteriors as described in section 3.3

2. calculation of $p(w|t_s, t_e, \mathbf{X})$ for all words (with associated times) found in the lattice. These constitute the initial clusters.
3. combination of overlapping clusters corresponding to the same word. Clusters are only considered for combination if they are not yet in (precedence) relation. As one clustering step can prevent other candidate cluster pairs from being allowed to combine, the candidate pairs are ordered according to their relative time overlap weighted by the posterior probabilities.
4. combination of remaining overlapping cluster (with different words) until a total order of the clusters is achieved. In this step the candidate pairs are ordered by the average phonetic similarity³ of the words in the respective clusters weighted by their posteriors:

$$dist(C_1, C_2) = \frac{1}{\|C_1\| \|C_2\|} \sum_{\{(w_1, w_2) | w_1 \in C_1, w_2 \in C_2\}} psim(w_1, w_2) p(w_1) p(w_2)$$

where $psim(w_1, w_2)$ is the phonetic similarity measure of the canonical pronunciations of the words w_1 and w_2 .

After the above algorithm has run until no further clustering candidates are available, the special deletion edges “-” are added. For each set C_i the sum of the posteriors is calculated and the remaining probability mass is assigned to “-” so that the posteriors sum to one. The reason why there is some probability mass available that has no been assigned to any of the normal words is that due to the nature of the clustering some links in the lattices are shifted (or rather squeezed) slightly forward or backward in time, i.e. not all links that span a specific time t (in the sense discussed in section 3.4) have to be assigned to the same cluster, some will be shifted to the previous or the following cluster.

Experimental results of an implementation of this clustering algorithm are discussed in 5.3.3.

An obvious shortcoming of the algorithm discussed here is the fact that it only aligns complete words. From examining the output of a real speech recogniser it is obvious that many of the errors are actually due to an incorrect (implicit) segmentation of the signal. For example often long words are actually recognised as a sequence of two or more shorter words that sound similar to the original word when concatenated. The opposite effect (i.e. merging two reference words into one hypothesis) is also seen quite frequently. By extending the clustering algorithm to take these phenomena into account

³based on a simple symmetric Levenshtein distance of their canonical pronunciations – this is an extremely simple metric as the similarity of phones is not taken into account. An improvement would be a phone based alignment that explicitly models the phonetic similarity of phones and the probability that particular phones are elided

the estimate of the expected word error rate could be improved. Hopefully the better alignment would lead to a further improvement in the 1-best rescoring result.

An alternative application of the confusion network clustering is presented in [Mangu and Brill, 1999], where the resulting network is used to prune the original word lattice. This is achieved by first pruning the confusion network to the required size (number of word sequences) and then intersecting it with the original lattice. Mangu reports significant improvements (in terms of oracle WER for a fixed size) over the conventional lattice pruning method.

The lattices generated by the clustering approach are also suited very well as a data structure at the interface between a speech recogniser and the higher processing components in a NLP system. The advantages for this application are that the confusion network contains a large number of hypotheses in a very compact form. Due to the use of posterior probabilities it is possible to provide the next processing step (e.g. Syntax) with accurate likelihood scores of each hypothesis without having to create a huge lattice with explicit time information and a duplication of links for language model purposes.

5 Experimental Results

To validate the rescoring approaches described in the previous two chapters, their performance was evaluated in the HTK continuous speech recognition system. The source code of the implementation of the rescoring techniques can be found in the appendix.

5.1 Description of Corpora

All the experiments were performed on two different continuous speech corpora:

hub4 This corpus consists of recordings of American Broadcast News. The interesting feature of this corpus is that it contains data from a wide variety of recording situations and speaking styles. Typically each sentence is assigned to one of seven *focus conditions* (see 5.1). Typically the recogniser accuracy depends strongly on the focus condition.

F0	baseline broadcast speech
F1	spontaneous broadcast speech
F2	speech over telephone channels
F3	speech in the presence of background music
F4	speech under degraded acoustical conditions
F5	non-native speakers
FX	all other speech

Figure 5.1: focus condition in Broadcast News

hub5 This corpus is a combination of the *CallHome English* and the *Switchboard* corpora and contains recordings of conversation made over the telephone. Due to the relatively bad recording conditions and the highly spontaneous nature of the utterances, the word error rate on this data is usually very high.

For an overview of the recognition systems used see the the descriptions in [Woodland et al., 1998a] and [Hain et al., 1999], respectively. For both corpora lattices were used that have been generated by a gender-dependent, MLLR adapted triphone systems using

a word based 4-gram language model smoothed with a class based trigram. The test sets on which the performance was assessed were the sets used in the 1997 Hub4 Broadcast News Evaluation and the 1998 Hub5 Switchboard Evaluation. The baseline 1-best error rate of the lattices used for rescoring are summarised in table 5.2.

	hub4	hub5
WER	17.4	43.0
SER	92.0	80.3

Figure 5.2: overall performance of the baseline system

Interestingly the sentence error rate of the hub4 corpus is slightly higher than for hub5 while the word error rate is about 2.5 times lower. This is an effect of the different length (in words) of the sentences (see the discussion in section 3.1 for more details).

In the following two different kinds of word error rate measurements will be used:

- The scores generated according to the official evaluation guidelines and transcriptions from NIST have been measured for hub4 and hub5.
- For hub4 in addition to this, scores based on the HTK scoring program were generated. The most important difference to the NIST scores is that the sentences used in the HTK scoring alignment are the same as the ones used for decoding. This guarantees that the quoted sentence error rate actually correspond to the relevant sentences as opposed to some rather arbitrarily defined “turn”. The HTK scores also do not include the elaborate mapping rules employed by NIST which define what kind of errors can be ignored. Therefore the HTK word error rates are always slightly higher than the corresponding NIST scores. For hub5 the NIST sentence error rates are accurate as the fixed segmentation into turns was used in the decoding.

5.2 Lattice Pruning

The first experiment run was based on the link posteriors estimated from the lattices as discussed in section 3.3. This experiment was run as a test of the implementation to verify that the estimated posteriors are sensible. The link posteriors were used to reduce the size of the lattices.

The conventional approach to pruning lattices is to calculate for each link the score of the best path passing through that link using a modified A*-search (see e.g. [Odell, 1995] or for a comparison with beam pruning in the decoder [Sixtus and Ortmanms, 1999]). When this score is more than a fixed threshold below the overall best path then the link is pruned. This method can significantly reduce the lattice size without degrading

the oracle word error rate too much because it employs scores which take the whole utterance into account.

The alternative pruning implemented here replaces these best-path scores by real posterior probabilities, i.e. the sum over all paths to the start node of the link (and from the end node, respectively).

The results of this experiment are summarised in the plot in figure 5.3, where the oracle word accuracy is plotted against the link density at different levels of pruning.

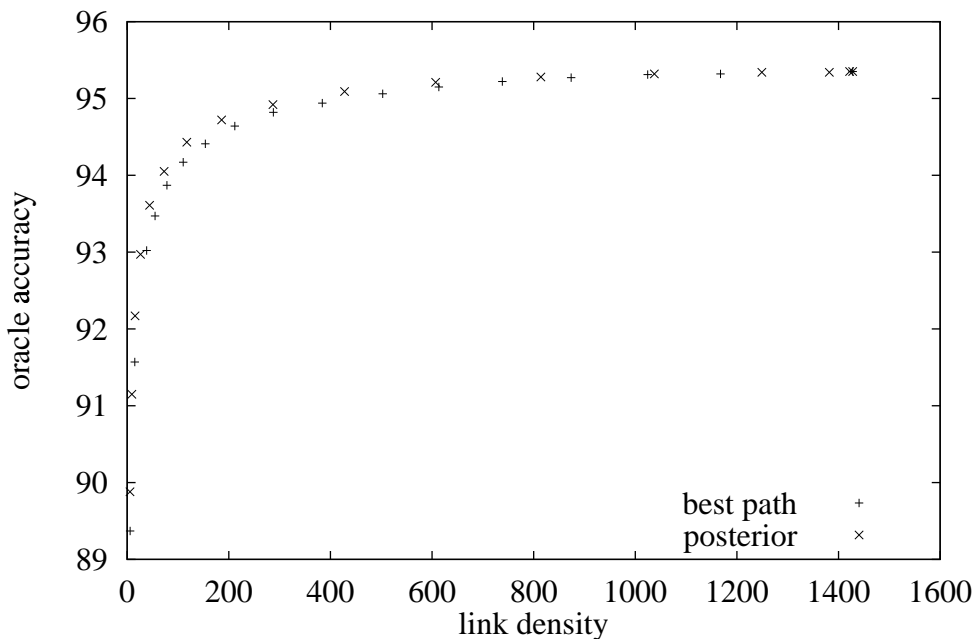


Figure 5.3: results of link posterior based lattice pruning on hub4

It can be seen that the posterior based pruning does not provide a significant difference from the conventional approach. This can be explained by the fact that for a given time there is a large number of links corresponding to the same word hypothesis (potentially with different boundary times). For the link posteriors these hypotheses are treated completely independently. A combination of these link posteriors to word posteriors as discussed in section 3.2 would probably improve these results.

5.3 Decoding

In this section the results produced with implementations of the three approaches to an optimised decoding strategy are examined.

5.3.1 N-best List Rescoring

The first set of decoding experiments used the N-best list based technique to minimise the word error rate described in section 4.1. The implemented decoder used the following decision function:

$$\hat{g}_{opt}(\mathbf{X}) = \underset{\mathbf{W} \in \mathcal{W}_{k-best}}{\operatorname{argmin}} E[WER(\mathbf{W}_i) | \mathbf{X}, \mathcal{W}_{n-best}] \quad (5.1)$$

To get a feeling for the expected word error rates estimated by this technique, 1000-best lists were generated from the hub4 lattices and $E[WER(\mathbf{W}_i) | \mathbf{X}, \mathcal{W}_{n-best}]$ was estimated for the first 100 hypothesis of some utterances. Plots of these values for two example sentences are shown in figure 5.4.

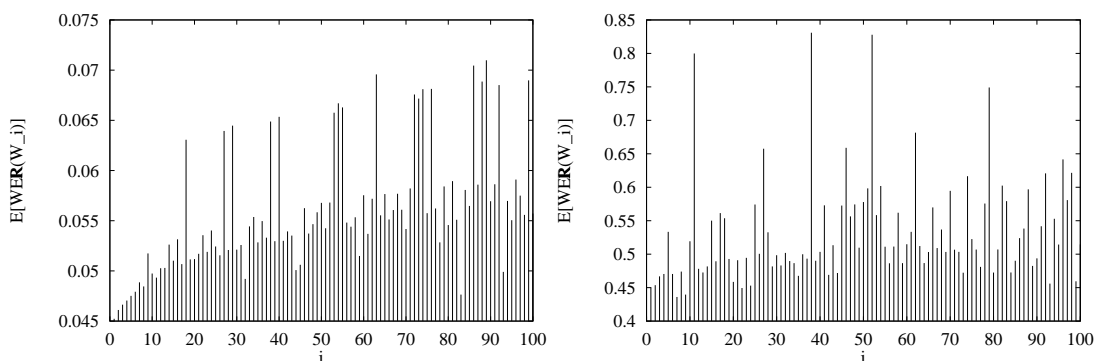


Figure 5.4: expected word error rate in N-best list

In the first utterance the 1-best hypothesis from the MAP decoder also has the lowest expected word error rate. In the second example the hypothesis that was only ranked seventh by the MAP decoder was optimal in the sense of equation 5.1. Overall the situation depicted in the left example seemed to be more typical. The case that the shape of the plot is relatively flat as in the right example was very rare.

Based on the relatively sharp rise in the word error rate as seen in the left example it seemed justified to restrict the number of hypotheses for which to calculate the word error to a small proportion of the N-best list, which provided the posterior estimates.

For both corpora it proved sufficient to consider the first 25 hypotheses and use an N-best list with 1000 entries for the posterior estimate (i.e. $k = 25$ and $n = 1000$ in equation 5.1).

In figure 5.5 histograms can be found that show from which ranks in the MAP N-best list the new optimal hypothesis was chosen. For each rank in the MAP N-best list (x-axis) the number of sentences where it was optimal is plotted. Here the counts for rank 1 (i.e. the $E[WER]$ decision agrees with the MAP decoder) are omitted as they are much larger than the counts of the other ranks combined and would thus distort the scale of the graphs. In the hub4 corpus for 528 sentences the MAP 1-best hypothesis was chosen (70%), for hub5 this was the case for 3138 sentences (72%).

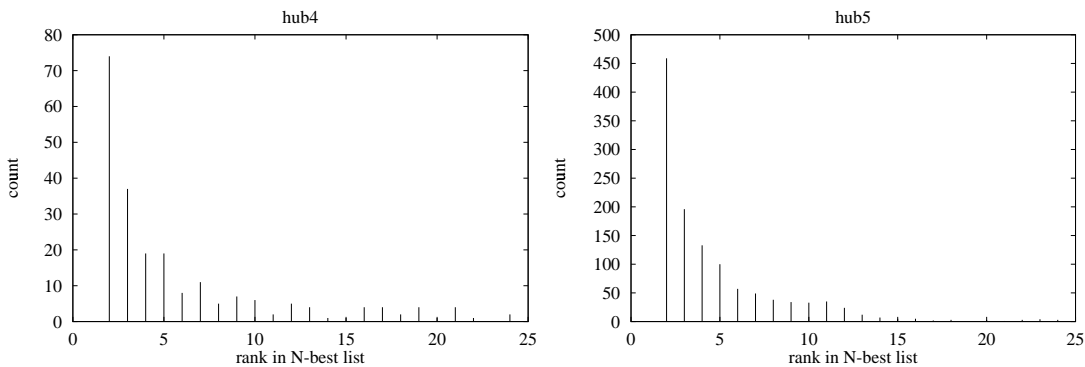


Figure 5.5: rank of optimal hypothesis in MAP N-best list

From the recognition results given in table 5.6 it can be seen that the N-best rescoring gives a small but consistent improvement over the standard MAP decoder on both corpora. The shift from the sentence based objective function in the MAP decoder to a word based objective function resulted in the expected decreased word error rate at the cost of an increased sentence error rate.

	hub4			hub5	
	WER(HTK)	WER(NIST)	SER	WER	SER
baseline	19.76	17.4	92.0	43.0	80.3
N-best	19.57	17.2	92.4	42.4	80.6

Figure 5.6: decoding results for N-best list rescoring

The amount of improvement found is roughly consistent with the results reported in [Stolcke et al., 1997] for the hub5 corpus (Stolcke et al. reported slightly below 1% relative WER reduction while we achieved just over 1%). Contrary to Stolcke's findings

we found that the technique is also efficient on a corpus that yields much lower overall error rates. Presumably this is an effect of the high sentence lengths found in the hub4 corpus, which tend to increase the discrepancy between the sentence and word error rates even if the overall performance is good.

5.3.2 Time Dependent Posteriors

In the next experiment the lattices were used to estimate link posteriors which were then combined to time dependent word posteriors as described in section 3.4. These were used as an additional likelihood factor in the objective function as indicated in equation 3.13. Preliminary experiments on the hub4 corpus showed that the results were very stable independent of the exponential weight used to integrate the posteriors into the objective function. Therefore it was decided not to use any weight at all but just multiply the MAP function directly with the posteriors.

	hub4			hub5	
	WER(HTK)	WER(NIST)	SER	WER	SER
baseline	19.76	17.4	92.0	43.0	80.3
posterior	19.39	17.0	92.0	43.0	80.7

Figure 5.7: decoding results for rescoring with time dependent posteriors

As can be seen from table 5.7 this technique resulted in another improvement over the N-best list rescoring described above on the hub4 corpus. Alas no significant improvement was achieved on the hub5 corpus. Table 5.8 shows the results (NIST WER) for the different focus conditions in the Broadcast News corpus.

focus	baseline	posterior
F0	10.4	10.3
F1	17.1	16.5
F2	21.7	20.7
F3	28.4	28.1
F4	20.9	20.7
F5	24.4	23.9
FX	32.6	32.6
avg.	17.4	17.0

Figure 5.8: WER results for different focus conditions of hub4

The improvement is fairly consistent over all focus conditions and not restricted to only the well recognised conditions such as prepared broadcast speech (F0). To transfer

these improvements to the hub5 data it would presumably be necessary to introduce a weighting factor for the posteriors and tune this factor on a separate development set containing similar data.

5.3.3 Confusion Networks

The final set of decoding experiments was performed with an implementation of the clustering algorithm described in section 4.3. For each utterance a confusion network was produced from the word lattice and the best path was extracted from these networks.

As this technique is in some sense a generalisation of the N-best list based word error rate minimisation, it was expected to produce superior results. These improvements are mainly due to the improved estimate of the posterior probabilities, which is now based on all paths in the lattice instead of just the 1000 best.

	hub4			hub5	
	WER(HTK)	WER(NIST)	SER	WER	SER
baseline	19.76	17.4	92.0	43.0	80.3
N-best	19.57	17.2	92.4	42.4	80.6
confnet	19.23	17.0	92.5	41.9	80.6

Figure 5.9: results of confusion network clustering

The confusion network based rescored indeed results in a further improvement over the N-best list based rescoring (see table 5.9 for the details). A problem with this approach is that due to the clustering of links it is no longer possible to exactly identify the start and end times of the word hypotheses. This does not have an impact on the word error rate measured with HTK, which aligns the hypothesis and reference word string on the basis of sentences. The NIST scoring on the other hand concatenates the sentences used in decoding and performs the alignment on larger chunks by taking the time information into account. This difference in scoring protocols explains the fact that the word error rate as measured by HTK is improved compared to the time dependent posterior approach but the error rate measured by the NIST software remains unchanged. Presumably the NIST word error rate could be improved by performing a forced alignment of the utterance with the new word sequence to generate accurate word boundary times.

The results of the confusion network clustering approach on the hub5 corpus were also very consistent over different subsets of the corpus as shown in table 5.10.

	baseline	confnet
female	44.6	43.6
male	40.2	38.8
Switchboard	40.6	39.1
Callhome	45.4	44.7

Figure 5.10: NIST WER results of confusion network clustering on hub5

5.3.4 Summary

The experiments described in the preceding three sections clearly demonstrate that a word based decoding function can be successfully employed to improve the recognition results in terms of the word error rate. The results are summarised in table 5.11.

	hub4			hub5	
	WER(HTK)	WER(NIST)	SER	WER	SER
baseline	19.76	17.4	92.0	43.0	80.3
N-best	19.57	17.2	92.4	42.4	80.6
posterior	19.39	17.0	92.0	43.0	80.7
confnet	19.23	17.0	92.5	41.9	80.6

Figure 5.11: summar of decoding results

It is especially important to note that the techniques are effective on the two very different corpora. The overall relative reduction of the word error rate achieved is 2.2% and 2.6% respectively.

5.4 Confidence Scores

As mentioned in section 3.4 the time dependent posteriors can be interpreted as confidence scores. The posteriors calculated in the decoding experiments were used to generate word confidence scores for the corresponding word sequences.

For a given word in the final hypothesis of the decoder, the geometric mean of the time dependent word posteriors over the duration of the word is used as the confidence score. No further processing or normalisation of these scores was performed. The results in terms of the Normalised Cross Entropy are given in table 5.12. The baseline results quoted were produced by the full HTK evaluation systems (i.e. the result of combining triphone and quinphone results, see [Woodland et al., 1998a]). In these systems the scores were produced by mapping N-best homogeneity score with decision trees into the range between zero and one.

	hub4	hub5
HTK baseline	0.179	0.182
posterior	0.302	0.106

Figure 5.12: normalised cross entropy of confidence scores

The poor performance of the posterior scores on the hub5 task is consistent with the lack of improvement in the corresponding rescoring experiment described in section 5.3.2. On the hub4 task a significant improvement over the N-best homogeneity based measures was achieved. This improvement is also illustrated by the DET-curves of the two approaches on the hub4 corpus shown in figure 5.13.

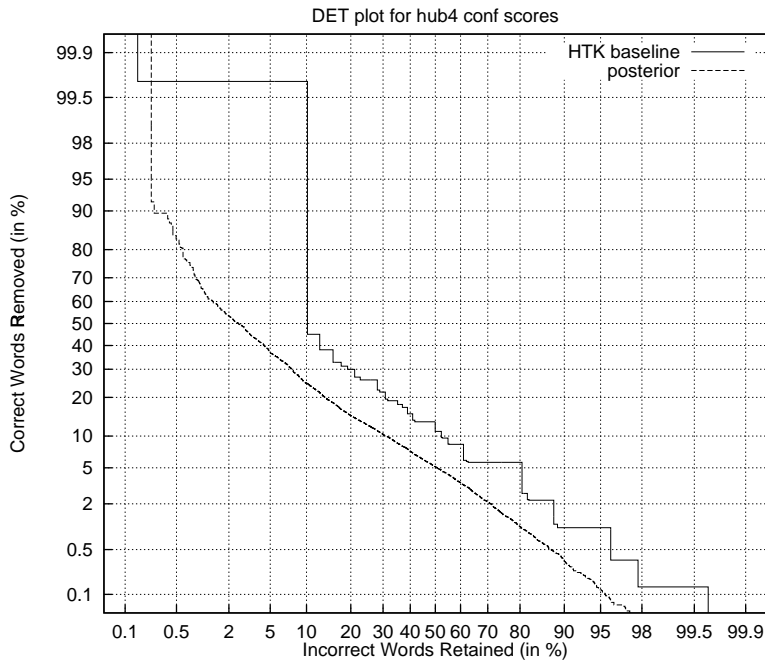


Figure 5.13: DET curves of the confidence scores for the hub4 testset

This curves visualise the tradeoff between the two kinds of errors which occur when confidence scores are used to reject poorly recognised words. Depending on the threshold chosen correct words are rejected (y-axis) or incorrect words are retained (x-axis). Further detail about the use of DET curves in speech recognition can be found in [Martin et al., 1997]. In general a better estimator of confidence scores results in a curve closer to the lower left corner of the plot. The posterior based estimates clearly outperform

the baseline system.

6 Conclusions & Further Work

It has been argued that the maximum a posteriori criterion traditionally employed in automatic speech recognition systems yields only suboptimal performance in terms of the word error rate. While from a theoretical standpoint the MAP decoder is optimal with respect to minimising the sentence error rate, it is not very well matched to the word error rate metric which is much more appropriate for many tasks.

Based on the word lattices generated by a MAP decoder a rescoring approach was implemented that aims to directly use the expected word error rate as the objective function in the decoding. An implementation of this technique was evaluated on two continuous speech corpora and achieved significantly better results than the conventional decoder. The concept of time dependent posteriors was introduced and an approach to their estimation presented. These posteriors were used successfully in lattice rescoring and confidence annotation.

Obvious areas for further extensions of the approach described in this thesis include the integration of the techniques developed in the fields of confidence scoring and key-word spotting into the rescoring paradigm. Especially acoustic features that have been successfully used for confidence annotation hold some promise as additional knowledge sources that can be employed to improve the estimation of word posteriors and thus lead to better decoding results. A somewhat related problem is tackled by the ROVER program developed by NIST (see [Fiscus, 1997]) that tries to integrate the hypotheses from a number of complete speech recognition system into one new hypothesis that is superior to all the individual systems. This hypothesis combination could clearly profit from reliable estimates of word posterior probabilities by the individual systems. These could be used to combine not only the 1-best word strings of the systems but instead essentially perform the combination on the basis of word (posterior) distributions to find the optimal recognition hypothesis.

The general problem of designing a decoder which is tuned to the particular application and situation it will be used in, is still largely unsolved. It seems clear that in the coming years this problem will become even more relevant as new and more ambitious applications of speech recognition become possible. As speech recognisers are integrated into more and more complex dialogue systems it will become necessary to investigate ways to optimise the decoder according to higher level metrics related to the overall system performance. An example of a first step into this direction is the work on tuning

a decoder for an Information Extraction task described in [Goel and Byrne, 1999].

Bibliography

- [Amtrup et al., 1997] Amtrup, J. W., Heine, H., and Jost, U. (1997). What's in a word graph? evaluation and enhancement of word lattices. In *Proc. Eurospeech*, pages 2663 – 2666.
- [Cox and Rose, 1996] Cox, S. and Rose, R. C. (1996). Confidence Measures for the SWITCHBOARD database. In *Proc. ICASSP'96*, volume 1, page 511, Atlanta, USA.
- [Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley, New York.
- [Fetter, 1997] Fetter, P. (1997). *Detection and Transcription of Out-Of-Vocabulary-Words*. PhD thesis, TU Berlin.
- [Fiscus, 1997] Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER). In *Proc. IEEE ASRU Workshop*, pages 347–352, Santa Barbara.
- [Fukunaga, 1990] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition.
- [Goel and Byrne, 1999] Goel, V. and Byrne, W. (1999). Task dependent loss functions in speech recognition: Application to named entity extraction. In *Proceedings of the ESCA Workshop on Spoken Audio Retrieval*.
- [Goel et al., 1998] Goel, V., Byrne, W. J., and Khudanpur, S. P. (1998). LVCSR rescoring with modified loss functions: A decision theoretic perspective. In *Proc. ICASSP98*, pages 425–428.
- [Hain et al., 1999] Hain, T., Woodland, P. C., Niesler, T. R., and Whittaker, E. W. (1999). The 1998 HTK system for transcription of conversational telephone speech. In *Proc. ICASSP99*.
- [Mangu and Brill, 1999] Mangu, L. and Brill, E. (1999). Lattice compression in the consensual post-processing framework. In *Proceedings SCI/SAI*.
- [Mangu et al., 1999] Mangu, L., Brill, E., and Stolcke, A. (1999). Finding consensus among words: Lattice-based word error minimization. In *Proc. Eurospeech*.

- [Martin et al., 1997] Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybicki, M. (1997). The DET curve in assessment of detection task performance. In *Proc. Eurospeech*.
- [Nádas, 1985] Nádas, A. (1985). Optimal solution of a training problem in speech recognition. *IEEE-ASSP*, 33(1):326–329.
- [Odell, 1995] Odell, J. (1995). *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University.
- [Ortmanns et al., 1997] Ortmanns, S., Ney, H., and Aubert, X. (1997). A word graph algorithm for large vocabulary speech recognition. *Computer Speech & Language*, 11(1):43–72.
- [Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.
- [Schaaf and Kemp, 1997] Schaaf, T. and Kemp, T. (1997). Confidence measures for spontaneous speech. In *Proc. ICASSP97*, pages 875–878.
- [Schukat-Talamazzini, 1995] Schukat-Talamazzini, E. G. (1995). *Automatische Spracherkennung*. Vieweg. (in German).
- [Siu et al., 1997] Siu, M., Gish, H., and Richardson, F. (1997). Improved estimation, evaluation and application of confidence measures for speech recognition. In *Proc. Eurospeech*.
- [Sixtus and Ortmanns, 1999] Sixtus, A. and Ortmanns, S. (1999). High quality word graphs using forward-backward pruning. In *Proc. ICASSP99*.
- [Stolcke et al., 1997] Stolcke, A., König, Y., and Weintraub, M. (1997). Explicit word error minimization in N-best list rescoring. In *Proc. Eurospeech*.
- [Weintraub, 1995] Weintraub, M. (1995). LVCSR Log-Likelihood Ratio Scoring for Keyword Spotting. In *Proc. ICASSP'95*, volume 1, page 297, Detroit, USA.
- [Weintraub et al., 1997] Weintraub, M., Beaufays, F., Rivlin, Z., König, Y., and Stolcke, A. (1997). Neural - network based measures of confidence for word recognition. In *Proc. ICASSP97*, pages 887–890.
- [Wessel et al., 1998] Wessel, F., Macherey, K., and Schlüter, R. (1998). Using word probabilities as confidence measures. In *Proc. ICASSP98*, pages 225–228.
- [Woodland et al., 1998a] Woodland, P., Hain, T., Johnson, S., Niesler, T., Tuerk, A., Whittaker, E., and Young, S. (1998a). The 1997 HTK broadcast news transcription system. In *Proceeding DARPA Broadcast News Transcription Workshop*.

[Woodland et al., 1998b] Woodland, P. C., Hain, T., Johnson, S. E., Niesler, T. R., Tuerk, A., and Young, S. J. (1998b). Experiments in broadcast news transcription. In *Proc. ICASSP98*, pages 909–912.