NTNU – Trondheim
Norwegian University of
Science and Technology

Spoken Document Classification of Broadcast News

Håkon Sandsmark

Master of Science in Communication Technology
Submission date: July 2012
Supervisor: Torbjørn Svendsen, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Abstract

Two systems for spoken document classification are implemented by combining an automatic speech recognizer with the two classification algorithms naive Bayes and logistic regression. The focus is on how to handle the inherent uncertainty in the output of the speech recognizer. Feature extraction is performed by computing expected word counts from speech recognition lattices, and subsequently removing words that are found to carry little or noisy information about the topic label, as determined by the information gain metric. The systems are evaluated by performing cross-validation on broadcast news stories, and the classification accuracy is measured with different configurations and on recognition output with different word error rates. The results show that a relatively high classification accuracy can be obtained with word error rates around 50%, and that the benefit of extracting features from lattices instead of 1-best transcripts increases with increasing word error rates.

Sammendrag

To systemer for emneklassifisering av tale er implementert ved å kombinere automatisk talegjenkjenning med de to klassifiseringsalgoritmene naiv Bayes og logistisk regresjon. Det fokuseres på hvordan den iboende usikkerheten i talegjenkjenningsresultatet kan håndteres. Egenskapsuttrekking gjøres ved å beregne forventet antall ordforekomster fra talegjenkjenningsgrafer («lattices»), for deretter å fjerne ord som gir liten eller forvirrende informasjon om emnet, som målt ved informasjonsgevinst. Systemene er evaluert ved bruk av kryssvalidering på nyhetsinnslag fra kringkastingsmedier, og klassifikasjonsnøyaktigheten er målt for forskjellige konfigurasjoner og på talegjenkjenningsresultater med forskjellige ordfeilrater. Resultatene viser at en relativt høy klassifikasjonsnøyaktighet kan oppnås med ordfeilrater på rundt 50%, og at fordelen ved å gjøre egenskapsuttrekkingen fra talegjenkjenningsgrafer i stedet for topprangerte transkripsjonshypoteser øker med økende ordfeilrate.

Preface

This thesis is submitted to the Norwegian University of Science and Technology in partial fulfillment of the requirements for the degree Master of Science. The work has been performed at the Department of Electronics and Telecommunications in the spring semester of 2012.

I would like to express my deep gratitude to my supervisor, Professor Torbjørn Svendsen, for his support and encouragement during the process.

In order to limit the scope of the thesis, a basic knowledge of probability theory, set theory, graph theory and linear algebra is assumed on behalf of the reader. I hope you enjoy the read.

Oslo, July 9, 2012
Håkon Sandsmark

Contents

— 1 —
Introduction

1.1 Definition and Motivation

With an ever-increasing amount of spoken data being produced, it becomes increasingly important to be able to properly organize such data in for example audio databases. A common organization task is categorization, or *classification*, into different topics. The term *spoken document classification* is defined as the problem of labeling a digital speech segment with the correct topic from a fixed set of possible topics.

A system for automatic spoken document classification can for example be used in applications such as classification of broadcast news stories into topics like "science" and "sports", and classification of conversations from a surveilled telephone line into either "criminal" or "innocent" activity.

Spoken document classification is closely related to spoken document *retrieval*, where a list of spoken documents is returned in response to a given query. This retrieval approach is suitable when the user has a very specific need to which only a relatively small number documents are relevant. The classification approach is better suited when the user is looking for documents matching a more general definition of a topic that is hard to express in form of a query. Classification and retrieval can also be combined to improve the user experience, for example to help disambiguate a query like `mine`, which could mean both an explosive device and an excavation from which minerals are extracted. If the user was able to filter the returned documents by topics like "war" and "economics", it could help the user find relevant documents more quickly.

1.2 Challenges

In many situations, there is no meta-data available to help with the classification, and the speech itself is then the only source of information. Doing topic classification manually can be very time-consuming, and this is especially true for a linear medium like audio where the ability to manually scan through a spoken document is limited.

In this thesis the focus is on *supervised learning*, which means that an automatic spoken document classifier will learn parameters from *training documents* manually labeled with topics. While some manual labeling is still required at first, the goal is that the classifier should learn the characteristics of the classes from the training documents such that it becomes able to classify previously unseen documents with high accuracy.

A spoken document classifier can be built by combining an automatic speech recognizer with a topic classification algorithm. Such a combination will consist of three different modules: the speech recognizer, a feature extractor, and the topic classifier. In this thesis, the speech recognizer is assumed to be given (except for parameter adjustments), and the focus is on how to perform feature extraction and classification of the speech recognition output.

An important question in such a system is how to handle the inherent uncertainty in the textual representation of speech. The analysis will focus on the impact of speech recognition errors on classification accuracy, and whether considering multiple hypotheses from the speech recognizer, instead of only the most probable hypothesis, can help increase accuracy. Feature selection will also be employed, which means that only a subset of the speech recognition vocabulary is considered for classification. The idea is that some words are thought to provide either little or noisy information about the topic label, such that removing them could be beneficial for classification accuracy.

1.3 Thesis Outline

The remainder of this thesis is organized as follows.

Chapter 2 covers necessary background theory for spoken document classification. First, automatic speech recognition is explained, including how multiple recognition hypotheses can be obtained in form of a lattice. Then, a formal definition of the classification problem is given, followed by coverage of feature extraction in general and from lattices in particular. Finally, feature selection and the two classification algorithms naive Bayes and logistic regression are covered. The chapter is rounded off by discussing previously proposed approaches to spoken document classification.

Chapter 3 explains the approach to spoken document classification taken in this work. A brief architectural overview is given, before implementation details about the speech recognizer, the feature extractor and the classifiers are given.

In Chapter 4, details about the broadcast news data set used for evaluation are given. The evaluation methodology and evaluation metrics are covered, before the speech recognition and classification results are reported. The results and their implications are discussed at the end of this chapter.

The conclusion is given in Chapter 5, along with proposals of work that can be done to further improve the classification accuracy. Some more general ideas about future approaches to spoken document classification and related fields are also discussed.

— 2 —
Theory & Previous Work

In this chapter we will cover some background theory for this thesis. First, automatic speech recognition is covered, before we continue by covering classification in general, and then spoken document classification in particular.

Let us begin by introducing some probability notation. We will use capital letters to denote random variables, lowercase letters to denote a particular value taken by a random variable, and blackboard bold to denote its sample space, i.e. $\sum_{x\in\mathbb{X}} P(X = x) = 1$ for any discrete random variable X taking values in the sample space $\mathbb{X}$. We will often use the shorthand notation $P(x)$ instead of $P(X = x)$.

2.1 Automatic Speech Recognition

For this project we will assume that the automatic speech recognition (ASR) system is given and already trained. The process of finding the N most probable transcription hypotheses is referred to as N-best recognition. If $N = 1$, we will explicitly write 1-best recognition.

We will in this section briefly describe how 1-best recognition can be performed by an ASR system, and then continue by discussing how N-best recognition can be performed, as we are interested in the consequence of considering more than only the 1-best recognition result for classification.

When we feed the ASR system with an acoustic feature sequence $\mathbf{o} = o_1 o_2 \dots o_T$, the 1-best recognition task is to find the word string $\hat{\mathbf{w}} = w_1 w_2 \dots w_L$ with the maximum posterior probability given that acoustic feature sequence. This maximum a posteriori estimate of the word string is given by

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{o}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{o}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{o})} = \arg\max_{\mathbf{w}} P(\mathbf{o}|\mathbf{w})P(\mathbf{w}), \tag{2.1}$$

where $P(\mathbf{o}|\mathbf{w})$ is given by our *acoustic model*, and $P(\mathbf{w})$ is given by our *language model*.

The acoustic modeling in modern ASR systems is typically based on hidden Markov models (HMMs) [1]. The probability $P(\mathbf{o}|\mathbf{w})$ may be modeled directly by word HMMs, but sub-word models such as phone models are more commonly used in large-vocabulary continuous speech recognition due to the lack of sufficient training data for words. Word models are then constructed by concatenating sub-word HMMs according to a pronunciation lexicon.

The most common language model $P(\mathbf{w})$ is a simple n-gram language model [2, Chapter 4], in which the probability of a word depends only on the $n - 1$ preceding

words:

$$
\begin{aligned}
P(\mathbf{w}) &= P(w_1 w_2 \dots w_L) \\
&= \prod_{i=1}^{L} P(w_i | \cap_{j=1}^{i-1} w_j) && \text{(by the chain rule)} \\
&= \prod_{i=1}^{L} P(w_i | \cap_{j=i-n+1}^{i-1} w_j) && \text{(by the } n\text{-gram model).}
\end{aligned}
$$

2.1.1 *Viterbi Search*

The Viterbi search is a dynamic programming algorithm for finding the most likely state sequence given an observation sequence. For speech recognition, the search is performed on a grammar network constructed by acoustic word units weighted according to the language model [3, Section 12.4]. An example of such a grammar network for a unigram language model is shown in Figure 2.1.

The algorithm works by keeping a record of the previous state for the most likely path ending in each state for each time step. The search is referred to as time-synchronous because the time steps are processed consecutively from 1 to T. The most likely state sequence is then found by backtracking from the state having the most likely path at time T. Usually we return a sub-optimal estimate of the *word sequence* $\hat{\mathbf{w}}$ taken directly from the most likely *state sequence*, and this is called the Viterbi approximation.

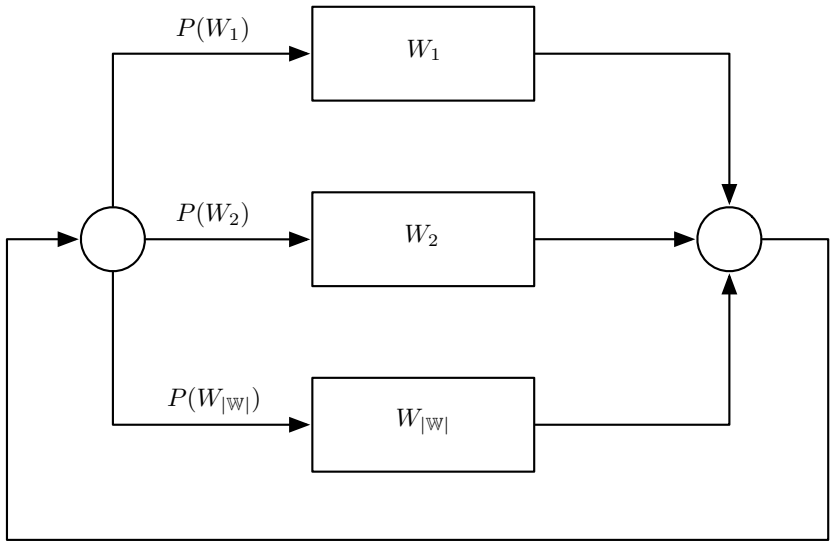


Figure 2.1: A grammar network for a unigram language model. Based on Figure 12.14 from [3].

The extension from a unigram language model to a bigram language model is done by letting the words have separate start and end nodes in order to "remember" the preceding word, as shown in Figure 2.2.

Extension beyond a bigram language model violates the first-order Markovian property that the Viterbi algorithm depends on, and therefore the search space has to be expanded to recover the first-order property. If the original search space (the vocabulary) is $\mathbb{W} = \{W_1, W_2, \ldots, W_{|\mathbb{W}|}\}$, and we have an n-gram language model with $n > 2$, then we need to expand the search space to $\mathbb{W}^{n-1}$ to recover the first-order property. This trick works because now the word history can be embedded in the current *grammar state word*, such that the transition probability from a grammar state word $(w_{i-n+1}, \ldots, w_{i-1}) \in \mathbb{W}^{n-1}$ to any grammar state word beginning with w_i is the n–gram probability $P(w_i|w_{i-1}, \ldots, w_{i-n+1})$. The exponential growth in the search space means that language models quickly become infeasible with increasing n.

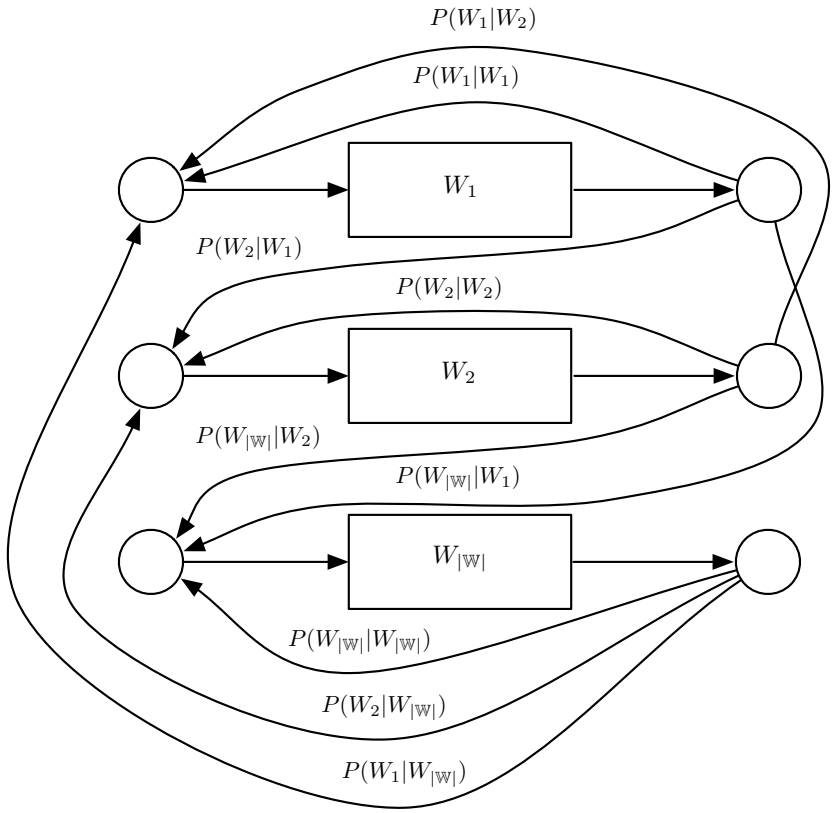


Figure 2.2: A grammar network for a bigram language model. Based on Figure 12.15 from [3].

2.1.2 *Pruning*

The full search space for the Viterbi search is $O(MT)$ and its time complexity is $O(M^2T)$ where M is the total number of HMM states after state expansion according to the language model [1] [3]. Since these numbers can become incredibly large for large-vocabulary speech recognition with higher-order language models, a full Viterbi search is often infeasible, and it is common to perform *pruning* of the search space [4]. Instead of processing each state in each time step, only the states with the most promising hypotheses are processed. This heuristic may very well result in the optimal state path not being found, but it allows us to trade-off between speed and accuracy.

A common pruning method is *beam pruning*, where a path is removed if its probability is below some threshold (the beam width) of the most likely path so far. This is illustrated in Figure 2.3. Since the Viterbi algorithm already keeps track of the probability of the most likely path ending at each state, beam pruning is only a matter of comparing these already computed values.

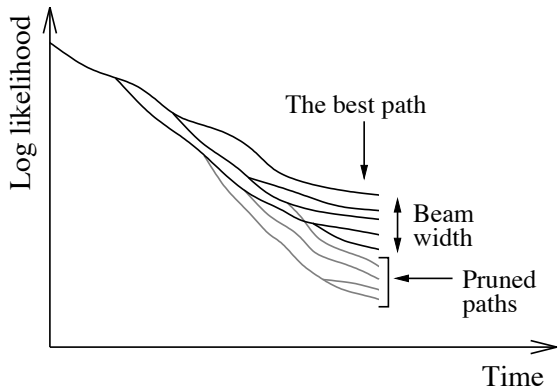


Figure 2.3: An illustration of beam pruning. Reproduced from [4].

2.1.3 *The N-Best Recognition Algorithm*

As previously discussed, we are interested in whether considering more than only the 1-best transcription will improve classification accuracy. The exact N-best algorithm [5] describes how to alter the time-synchronous Viterbi search to obtain an N-best list. The key difference is that the N-best algorithm keeps up to N separate records in each state and time step for paths that represent different word histories, where the standard Viterbi algorithm only keeps one record representing the path with the maximum probability ending in that state. If several paths arrive at the same state in the same time slot, and they have the same word history, their probabilities are added. If their word histories differ

and cause the number of records in that state to exceed N, only the records representing the N most likely word histories are kept. The final N-best list is found from the N records representing the most likely word histories at time T. Beam pruning can easily be applied to the N-best search, and the only difference from 1-best beam pruning is that it involves N times as many pruning candidates.

2.1.4 *Word Lattices*

A word lattice is in principle a compressed lossless representation of an N-best list of recognition hypotheses in which words overlapping in time in multiple hypotheses are merged. We represent a word lattice as a directed acyclic graph, where each vertex (node) is labeled with a word, and each edge is labeled with a joint language and acoustic probability. Each path through the lattice represents a recognition hypothesis consisting of the words from the vertices on the path, with a joint acoustic and language probability $P(\mathbf{o}|\mathbf{w})P(\mathbf{w}) = P(\mathbf{o}, \mathbf{w})$, as seen in Equation 2.1, given by the product of the edge probabilities on the path.

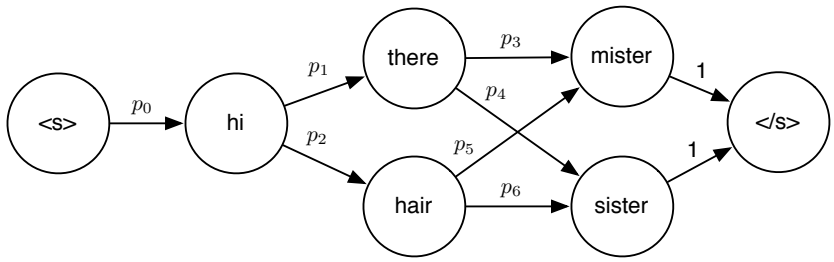


Figure 2.4: An example of a word lattice containing four different recognition hypotheses. The probability of a hypothesis can be calculated by multiplying the edge probabilities on its path.

2.1.5 *Evaluation*

The usual evaluation metric for an end-to-end automatic speech recognition system is the word error rate (WER). The WER is the word edit distance between the reference string and the recognized string, divided by the number of words in the reference string. The word edit distance is defined as the minimum total number of insertions, deletions and substitutions of words needed to transform the recognized string to the reference string. Then we have

$$\text{WER} = \frac{\text{number of insertions} + \text{number of deletions} + \text{number of substitutions}}{\text{number of words in reference}}.$$

If the reference word string is $v_1 v_2 \dots v_m$ and the recognized word string is $w_1 w_2 \dots w_n$, then the edit distance is $d_{m,n}$, given by the recursive definition

$$
\begin{aligned}
&d_{0,0} = 0 \\
&d_{i,0} = i,\ 1 \leq i \leq m \\
&d_{0,j} = j,\ 1 \leq j \leq n \\
&d_{i,j} = \min\left[d_{i-1,j} + 1,\ d_{i,j-1} + 1,\ d_{i-1,j-1} + \mathbf{1}_{v_i \neq w_j}\right],\ 1 \leq i \leq m,\ 1 \leq j \leq n,
\end{aligned}
$$

where $\mathbf{1}_{v_i \neq w_j}$ is an indicator function taking the value one if the subscript condition is true, and zero otherwise. The first argument to $\min$ represents a deletion, the second represents a insertion, and the third represents a substitution if $v_i \neq w_j$ or a match if $v_i = w_j$. The edit distance $d_{m,n}$ can be calculated in time $O(mn)$ by using dynamic programming to construct an $(m+1) \times (n+1)$ matrix containing the elements $d_{i,j}$.

2.2 Classification Overview

We will begin this section by giving a formal definition of a classifier and a supervised learning algorithm.

A classifier is a function $\gamma : \mathbb{X}^D \to \mathbb{C}$ that maps an instance $\mathbf{x} \in \mathbb{X}^D$ to a class $c \in \mathbb{C}$. The D-dimensional instance space $\mathbb{X}^D$ is called the *feature space*, in which each feature takes values from the set $\mathbb{X}$, which is assumed to be the real numbers $\mathbb{R}$ or some subset like the natural numbers $\mathbb{N}$ or $\{0, 1\}$. A single instance $\mathbf{x}$ is called a *feature vector*. A supervised learning algorithm is a function Γ that takes a training set $\mathbb{D} \subset \mathbb{C} \times \mathbb{X}^D$ as input and returns a learned classifier $\Gamma(\mathbb{D}) = \gamma$.

2.2.1 Feature Extraction

Feature extraction is the procedure of converting original source data to feature vectors. Domain knowledge about each specific problem is used to create a feature extractor that does this conversation. For text classification, the feature vector often follows the *bag-of-words model*, in which a document is represented by an unordered collection of words. Since the position of each word is ignored, we can use a simple feature vector representation in which each dimension corresponds to a word in the vocabulary. The value of each dimension (i.e. word) is usually the number of occurrences of that word in the document.

Word Counting in Spoken Documents

If we want to use the number of occurrences of each word, also called *word counts*, as features for a spoken document, we have to decide on how to calculate these numbers. We can count the number of occurrences of each word in the 1-best ASR transcription, or we can calculate *expected word counts* from the ASR lattice representation. We will now describe how the computation of expected word counts can be done.

We formally define a lattice as a directed acyclic graph with vertices (nodes) $\mathbb{V}$ and edges $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$. Each vertex $v \in \mathbb{V}$ is labeled with a word $w(v)$, and each edge $e = (u, v) \in \mathbb{E}$ is labeled with a probability $P(e)$ that is a joint language and acoustic probability. There is also a unique start vertex v_S and a unique end vertex v_E, for which there are defined two special words $w(v_S) =$ <s> and $w(v_E) =$ </s>.

The expected number of occurrences $c(w)$ for a word w can be calculated by summing the posterior probabilities for vertices labeled with that word, given by

$$\mathbb{E}\{c(w)|\mathbf{o}\} = \sum_{v \in \mathbb{V}: w(v)=w} P(v|\mathbf{o}),$$

where $P(v|\mathbf{o})$ is the posterior probability for a vertex v to be visited on a random path through the lattice.

Now, the problem of computing $P(v|\mathbf{o})$ is actually very similar to the problem of computing the probability of being in a particular HMM state at a specific time given an observation sequence. This HMM problem is solved efficiently by the forward-backward algorithm as shown by Rabiner in [1]. We will here derive a variant of the forward-backward algorithm for lattices.

A lattice path π is a sequence of consecutive edges in the lattice. Let $\mathbf{o}(\pi)$ denote the observation sequence corresponding to the path π, and let $u \rightsquigarrow v$ denote the set of all paths from vertex u to vertex v. We then define the forward variable $\alpha(v)$ for a vertex v as the probability mass of all partial paths from the start vertex ending in that vertex, given by

$$\alpha(v) = \sum_{\pi \in v_S \rightsquigarrow v} P(\pi, \mathbf{o}(\pi)) = \sum_{\pi \in v_S \rightsquigarrow v} \prod_{e \in \pi} P(e), \quad v \in \mathbb{V} \setminus \{v_S\}.$$

Similarly, the backward variable $\beta(v)$ for a vertex v is defined as the probability mass of all partial paths from that vertex to the end vertex, given by

$$\beta(v) = \sum_{\pi \in v \rightsquigarrow v_E} P(\pi, \mathbf{o}(\pi)) = \sum_{\pi \in v \rightsquigarrow v_E} \prod_{e \in \pi} P(e), \quad v \in \mathbb{V} \setminus \{v_E\}.$$

We also define that $\alpha(v_S) = \beta(v_E) = 1$.

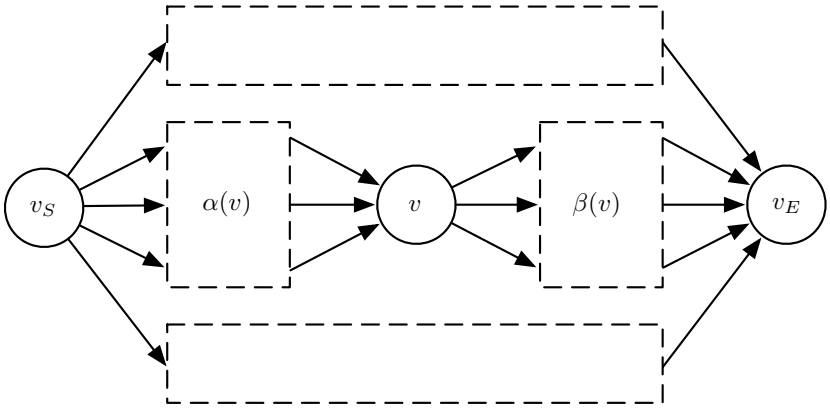


Figure 2.5: An illustration of the forward and backward variables for vertex v. The dashed boxes on the top and on the bottom represent the probability mass of the paths not passing through v.

We can now express $P(v|\mathbf{o})$ in terms of forward and backward variables:

$$\begin{aligned} P(v|\mathbf{o}) &= \sum_{\pi_1 \in v_S \rightsquigarrow v, \pi_2 \in v \rightsquigarrow v_E} P(\pi_1 \pi_2 | \mathbf{o}) \\ &= \frac{\sum_{\pi_1 \in v_S \rightsquigarrow v, \pi_2 \in v \rightsquigarrow v_E} P(\pi_1 \pi_2, \mathbf{o})}{P(\mathbf{o})} \\ &= \frac{\left(\sum_{\pi_1 \in v_S \rightsquigarrow v} P(\pi_1, \mathbf{o}(\pi_1))\right) \left(\sum_{\pi_2 \in v \rightsquigarrow v_E} P(\pi_2, \mathbf{o}(\pi_2))\right)}{\sum_{\pi \in v_S \rightsquigarrow v_E} P(\pi, \mathbf{o})} \\ &= \frac{\alpha(v)\beta(v)}{\alpha(v_E)}. \end{aligned}$$

Notice that the normalization factor in the denominator will ensure that the sum of the posterior probabilities of all lattice paths will be one. In other words, this is a posterior probability *conditioned on the lattice*, since any probability mass from paths pruned during recognition is ignored.

The only thing left now is the computation of the forward and backward variables. We will exploit the fact that they can be expressed by recursion:

$$\begin{aligned} \alpha(v_S) &= \beta(v_E) = 1 \\ \alpha(v) &= \sum_{u \in \mathbb{V}:(u,v) \in \mathbb{E}} \alpha(u) P((u,v)), \quad u \in \mathbb{V} \setminus \{v_S\} \\ \beta(u) &= \sum_{v \in \mathbb{V}:(u,v) \in \mathbb{E}} P((u,v)) \beta(v), \quad v \in \mathbb{V} \setminus \{v_E\}. \end{aligned}$$

Here the meaning behind their names is revealed since, for efficiency, the forward and backward variables should be calculated by dynamic programming for the vertices in forward and backward topological order, respectively. If they instead were calculated

in the opposite direction (directly from the recursion), it would result in a lot of redundant calculations.

2.2.2 *Feature Selection*

Feature selection is the process of selecting a subset of the original features with the purpose of using only those for training and classification. In other words, it is a form of dimensionality reduction of the original feature vector representation. For the bag-of-words representation, feature selection is equivalent to reducing the size of the vocabulary. In spoken document classification, this amounts to selecting a subset of the *ASR vocabulary* as the *classification vocabulary*.

The most obvious advantage of feature selection is that it makes learning and classification more computationally efficient. However, it can also help increasing the classification accuracy because it can remove noisy features. Consider for example the case when a rare word like `arachnocentric`[1] happens to only occur in documents from the class "sports" in the training data. If the classifier exaggerates the importance of this and tends to misclassify any document containing `arachnocentric` as "sports", the classifier has learned an accidental property of the training data, instead of the underlying characteristics, and this is called *overfitting*.

Information Gain Feature Selection

One method of performing feature selection is by ranking the features by their *information gain* [7] with respect to a training set, and then selecting a certain number features from the top of this ranking. The information gain of feature i, with a value of X_i, is defined as

$$IG(C|X_i) = H(C) - H(C|X_i),$$

where $H(C)$ is the entropy of the class label C, and $H(C|X_i)$ is the conditional entropy of the class label C given the feature value X_i. Intuitively, $IG(C|X_i)$ is the reduction in uncertainty about the class label C obtained by knowing the feature value X_i. In other words, it measures how much the feature helps with making the correct classification decision.

(1) Thanks to [6] for the example.

The information gain can be calculated as follows:

$$\begin{aligned} IG(C|X_i) &= -\sum_{c\in\mathbb{C}} P(c)\log P(c) + \sum_{x_i\in\mathbb{X}} P(x_i)H(C|x_i) \\ &= \sum_{c\in\mathbb{C}}\sum_{x_i\in\mathbb{X}} P(c,x_i)\log\frac{1}{P(c)} + \sum_{c\in\mathbb{C}}\sum_{x_i\in\mathbb{X}} P(x_i)P(c|x_i)\log P(c|x_i) \\ &= \sum_{c\in\mathbb{C}}\sum_{x_i\in\mathbb{X}} P(c,x_i)\log\frac{P(c,x_i)}{P(c)P(x_i)}. \end{aligned} \tag{2.2}$$

For the illustrative special case of binary classification and binary features (i.e. presence/absence of words), we have $\mathbb{C} = \mathbb{X} = \{0,1\}$, and maximum likelihood estimation for a given word results in

$$\begin{aligned} IG(C|X_i) = &\frac{N_{00}}{N}\log\frac{N_{00}N}{N_{0*}N_{*0}} + \frac{N_{01}}{N}\log\frac{N_{01}N}{N_{0*}N_{*1}} + \\ &\frac{N_{10}}{N}\log\frac{N_{10}N}{N_{1*}N_{*0}} + \frac{N_{11}}{N}\log\frac{N_{11}N}{N_{1*}N_{*1}}, \end{aligned}$$

where N_{cx_i} is the number of instances with $C = c$ and $X_i = x_i$, $*$ is a wildcard matching any value, and N is the total number of instances.

χ^2 *Feature Selection*

Another way of performing feature selection is by ranking the features by the χ^2 statistic between X_i and C. The χ^2 statistic measures the lack of independence between the variables, and the intuitive motivation is that good features should be highly dependent on the class label. We refer to [7] for details about the calculation of the χ^2 statistic.

Discretization

The calculation of the information gain in Equation 2.2 requires $\mathbb{X}$ to be discrete set, and the same is true for calculating the χ^2 statistic. This is a problem for our continuous-valued expected word counts, so we need a way to discretize these values. One method of doing such discretization is proposed in [8]. The idea is to first sort all the training instances $\mathbb{D}$ according to their value for the feature to be discretized, before potential cut points are identified at positions in the sorted list where the class label changes. Each potential cut point T splits the data set into two subsets, $\mathbb{D}_1$ with values $\leq T$ and $\mathbb{D}_2$ with values $> T$, and then the *class entropy of the cut* is calculated as

$$H(T;\mathbb{D}) = \frac{|\mathbb{D}_1|}{|\mathbb{D}|}H(\mathbb{D}_1) + \frac{|\mathbb{D}_2|}{|\mathbb{D}|}H(\mathbb{D}_2),$$

where the entropy of a document set is defined as the entropy of its internal class distribution, with a maximum likelihood estimate of

$$H(\mathbb{D}) = -\sum_{c \in \mathbb{C}} \frac{N_c}{N} \log \frac{N_c}{N},$$

where N_c is the number of documents with class label c.

The potential cut point with the lowest class entropy is then chosen as the discretization threshold. This procedure continues recursively on each subinterval until all documents belonging to the subinterval either have the same class label or the same value, or if a stopping criterion is met. The stopping criterion proposed in [8] is a *minimum description length* stopping criterion. We refer to the paper for a description of this criterion, which is also based on information theory.

2.2.3 Evaluation

We want to evaluate the ability of a classifier to correctly predict future data. Therefore, it is essential to have disjoint *training data* and *testing data*, such that testing is performed on data not seen during training. A commonly used method is cross-validation, in which several independent trials (folds) are performed in a way such that each instance is used for both training and testing, but never in the same fold.

In k-fold cross-validation, the data set $\mathbb{D}$ is split evenly into k disjoint subsets $\mathbb{D}_1, \ldots, \mathbb{D}_k$. In fold i, subset $\mathbb{D}_i$ is used as test data and $\mathbb{D} \setminus \mathbb{D}_i$ is used as training data. The testing and training for each fold is done independently from all the other folds, and the evaluation results obtained in these independent tests are usually averaged across folds.

A common evaluation metric for multiclass classification is *accuracy*, which is defined as the fraction of correctly classified instances.

The cross-validation procedure can also be *stratified*, which is a way of performing the data split that ensures that each fold has a distribution of classes that matches the distribution of the data set as a whole. An example of the contrary (non-matching distributions) is a subset consisting solely of a single class, which obviously would cause problematic results.

The stratification is implemented by handling the instances belonging to each class separately. A total of $k|\mathbb{C}|$ disjoint subsets are formed by splitting the instances belonging to each class evenly into N disjoint subsets. The final k stratified folds are then obtained by combining these subsets such that each fold has one of these subsets from each class.

2.3 Classification Models & Algorithms

2.3.1 Naive Bayes

Naive Bayes is a probabilistic classifier in which the probability of a feature vector $\mathbf{x}$ belonging to class c is computed by using Bayes' rule:

$$P(c|\mathbf{x}) = \frac{P(c)P(\mathbf{x}|c)}{P(\mathbf{x})}.$$

The feature vector probability $P(\mathbf{x}|c)$ can be decomposed by the chain rule as

$$P(\mathbf{x}|c) = \prod_{i=1}^{D} P(x_i|c, x_{i-1}, \ldots, x_1).$$

To be able to calculate this probability, we need all the probabilities in the product. However, a table listing the distribution $P(x_i|c, x_{i-1}, \ldots, x_1)$ for all i would contain $|\mathbb{C}| \sum_{i=1}^{D} 2^i = |\mathbb{C}|2^{D-1}$ probabilities if the features are binary. This is clearly problematic because of the memory consumption of this table, but also because a model with this number of parameters is prone to overfitting. To avoid this problem, we employ the *naive Bayes assumption*, which states that each feature is conditionally independent of all the other features given the class label. Then we have that $P(x_i|c, x_{i-1}, \ldots, x_1) = P(x_i|c)$ for all i, and we have reduced the number of probabilities used for modeling $P(\mathbf{x}|c)$ for binary features from $|\mathbb{C}|2^{D-1}$ to $|\mathbb{C}|D$.

Classification with the naive Bayes model is performed by selecting the maximum a posteriori (MAP) estimate of the class:

$$\hat{c} = \arg\max_{c \in \mathbb{C}} P(c|\mathbf{x}) = \arg\max_{c \in \mathbb{C}} \frac{P(c)P(\mathbf{x}|c)}{P(\mathbf{x})} = \arg\max_{c \in \mathbb{C}} P(c) \prod_{i=1}^{D} P(x_i|c).$$

We can drop the denominator since it is fixed with respect to the class label, but in addition to modeling $P(\mathbf{x}|c)$, we also need to model the class priors $P(c)$.

Event Models

There are two popular variations of the naive Bayes model as applied to text classification [9]. The *Bernoulli model* has binary feature values indicating absence or presence of words, so it can be seen as generating the vocabulary for each document. For a binary feature vector representation with $\mathbb{X} = \{0, 1\}$, the probability of a class c is given by

$$P(c|\mathbf{x}) \propto P(c) \prod_{i=1}^{D} \left(x_i P(X_i = 1|c) + (1 - x_i)(1 - P(X_i = 1|c))\right),$$

where the model parameter $P(X_i = 1|c)$ is the probability of word i being present in a document from class c.

This Bernoulli model is simple, but problematic for our expected word counts where presence and absence of words cannot be directly defined.

The *multinomial model* instead has numerical features representing the number of occurrences of each word, and it can be seen as generating a document by selecting a word for each position in the document. The multinomial model has also been shown experimentally to give better accuracy than the Bernoulli model on many data sets unless the vocabulary is very small [9]. The probability of a class c given a feature vector $\mathbf{x}$ with word counts $x_i = c(w_i)$ is given by

$$P(c|\mathbf{x}) \propto P(c) \prod_{i=1}^{D} P(w_i|c)^{x_i},$$

where the model parameter $P(w_i|c)$ is the probability of word i appearing at a specific position in a document with class label c.

We notice that the Bernoulli model explicitly models absence of words, while the multinomial model will only take into account the words that are present in the document, since words with zero counts will result in factors of one in the probability calculation.

It is interesting to note that the multinomial naive Bayes model is equivalent to using a unigram language model for classification, in which each class is treated as its own language, and the most probable class for a document is given by the language model that has the highest probability of the document.

In this thesis, we will focus on the multinomial model, but we will turn to the Bernoulli model once to simplify a theoretical analysis.

Training

As a supervised classification model, naive Bayes learns its parameters from a set of training data. Maximum likelihood estimation of the parameters for the multinomial model is given by

$$\hat{P}(w|c) = \frac{N_{cw}}{\sum_{w' \in \mathbb{W}} N_{cw'}},$$

and the maximum likelihood estimator for the class priors is simply

$$\hat{P}(c) = \frac{N_c}{N},$$

where N_{cw} is the number of occurrences of the word w in documents with the class label c in the training data, N_c is the number of documents from class c in the training data, and N is the total number of training documents.

These estimates would give a probability of zero to a word not seen in the class during training, and since the probabilities are multiplied, this would result in the product being zero. To avoid this, we employ *Laplace smoothing*, which distributes some of the probability mass from words seen during training to words not seen during training [10]. This corresponds maximum a posteriori estimation with a uniform Dirichlet prior, which is the conjugate prior of the multinomial distribution. The expected posterior probabilities are then given by

$$\hat{P}(w|c) = \frac{N_{cw} + 1}{\sum_{w' \in \mathbb{W}} (N_{cw'} + 1)} = \frac{N_{cw} + 1}{(\sum_{w' \in \mathbb{W}} N_{cw'}) + |\mathbb{W}|}.$$

Decision Boundary

Now, for a theoretical analysis, let us consider the Bernoulli naive Bayes model for binary classification with $\mathbb{C} = \{c_1, c_2\}$. If we define

$$\theta_1 = \begin{pmatrix} \ln \frac{P(X_1=1|c_1)}{P(X_1=1|c_2)} \\ \vdots \\ \ln \frac{P(X_D=1|c_1)}{P(X_D=1|c_2)} \end{pmatrix} \text{ and } \theta_0 = \begin{pmatrix} \ln \frac{P(X_1=0|c_1)}{P(X_1=0|c_2)} \\ \vdots \\ \ln \frac{P(X_D=0|c_1)}{P(X_D=0|c_2)} \end{pmatrix},$$

then the *log-odds ratio* for the naive Bayes classifier can be expressed as

$$\begin{aligned} \ln \frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})} &= \ln \frac{P(c_1)P(\mathbf{x}|c_1)}{P(c_2)P(\mathbf{x}|c_2)} \\ &= \ln \frac{P(c_1)}{P(c_2)} + \sum_{i=1}^{D} \ln \frac{P(x_i|c_1)}{P(x_i|c_2)} \\ &= \ln \frac{P(c_1)}{P(c_2)} + \sum_{i=1}^{D} \left(x_i \ln \frac{P(X_i = 1|c_1)}{P(X_i = 1|c_2)} + (1 - x_i) \ln \frac{P(X_i = 0|c_1)}{P(X_i = 0|c_2)} \right) \\ &= \ln \frac{P(c_1)}{P(c_2)} + \theta_0^{\mathrm{T}} \mathbf{1} + (\theta_1 - \theta_0)^{\mathrm{T}} \mathbf{x}, \end{aligned}$$

which is a linear function in $\mathbf{x}$. The log-odds ratio can also be interpreted as a *decision boundary* since we decide on the class c_1 if and only if the log-odds ratio is greater than zero. So the Bernoulli naive Bayes classifier is linear in log space, and this is in fact true for naive Bayes in general [6, Section 14.4]. It might seem optimistic to hope for linear separability of the classes in real-word data, but with the relatively high dimensional feature vectors that are used in document classification, it turns out this is often the case [11].

2.3.2 Logistic Regression

Logistic regression is a probabilistic classifier that differs from naive Bayes by modeling the probability $P(c|\mathbf{x})$ directly, which is sufficient for classification. The logistic regression

classifier for binary classification with $\mathbb{C} = \{c_1, c_2\}$ can be derived by assuming a linear log-odds ratio with respect to the feature vector:

$$\ln \frac{P(c_1|\mathbf{x})}{P(c_2|\mathbf{x})} = \mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}},$$

where we have added an intercept (bias) term to the feature vector to obtain $\tilde{\mathbf{x}} = \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}}$. We can derive the posterior probability $P(c_1|\mathbf{x})$ from the log-odds ratio by exponentiating to get

$$\begin{aligned} \frac{P(c_1|\mathbf{x})}{1 - P(c_1|\mathbf{x})} &= \exp(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}}) \\ P(c_1|\mathbf{x}) &= \frac{\exp(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}})}{\exp(\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}}) + 1} = \frac{1}{1 + \exp(-\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}})}, \end{aligned}$$

which is the *sigmoid function* with parameter $\mathbf{w}^{\mathrm{T}}\tilde{\mathbf{x}}$.

For multiclass classification with the classes $\mathbb{C} = \{c_1, \ldots, c_{|\mathbb{C}|}\}$, the probability of a class $c_i \neq c_{|\mathbb{C}|}$ is

$$P(c_i|\mathbf{x}) = \frac{\exp(-\mathbf{w}_i^{\mathrm{T}}\tilde{\mathbf{x}})}{1 + \sum_{j=1}^{|\mathbb{C}|-1} \exp(-\mathbf{w}_j^{\mathrm{T}}\tilde{\mathbf{x}})},$$

and the probability of the last class $c_{|\mathbb{C}|}$ is such that the they sum to one:

$$P(c_{|\mathbb{C}|}|\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{|\mathbb{C}|-1} \exp(-\mathbf{w}_j^{\mathrm{T}}\tilde{\mathbf{x}})}.$$

Training

The training procedure for logistic regression consists of finding the $(D+1) \times (|\mathbb{C}| - 1)$ weight matrix $\mathbf{W} = \begin{pmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_{|\mathbb{C}|-1} \end{pmatrix}$ that maximizes the conditional likelihood of the training data. The likelihood, assuming N independently generated training instances $\mathbb{D} = \{(c^{(1)}, \mathbf{x}^{(1)}), \ldots, (c^{(N)}, \mathbf{x}^{(N)})\}$, is given by

$$\mathcal{L}(\mathbf{W}) = \prod_{i=1}^{N} P(c^{(i)}|\mathbf{x}^{(i)}; \mathbf{W}),$$

and the maximum likelihood estimate of the weights is then given by

$$\begin{aligned} \hat{\mathbf{W}} &= \arg\max_{\mathbf{W}} \log \mathcal{L}(\mathbf{W}) \\ &= \arg\max_{\mathbf{W}} \sum_{i=1}^{N} \log P(c^{(i)}|\mathbf{x}^{(i)}; \mathbf{W}). \end{aligned}$$

We can use indicator functions to write this out in terms of posterior probabilities for all the classes:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \sum_{i=1}^{N} \sum_{j=1}^{|\mathbb{C}|} \mathbf{1}_{c^{(i)}=c_j} \log P(c_j|\mathbf{x}^{(i)}; \mathbf{w}_j).$$

There is however no closed-form solution to this maximization, due to the non-linearity of the sigmoid function. Luckily, this maximization problem is convex [12, Section 4.3], so any local optimum is also a global optimum, and therefore gradient ascent or Newton's method with random initial weights can be used to approach the global maximum.

2.3.3 *Discriminative and Generative Models*

A generative classifier learns the joint probability $P(\mathbf{x}, c)$, and can then estimate the posterior probability $P(c|\mathbf{x})$ of a given observation $\mathbf{x}$ by using Bayes' rule. The maximum a posteriori (MAP) estimate is the class c that maximizes this probability. Naive Bayes is an example of such a classifier. The classifier is called generative since it can generate new data from the learned joint distribution. One application exploiting this property of generative models is HMM-based speech synthesis [13]. Since HMMs are generative models, they can be used for generation of speech, in addition to recognition of speech as explained previously in this chapter.

A discriminative classifier instead learns the posterior probability $P(c|\mathbf{x})$ directly, which is sufficient for classification. This is what logistic regression does. A theoretical argument for this approach is that "one should solve the [classification] problem directly and never solve a more general problem as an intermediate step [such as modeling $p(\mathbf{x}|c)$]" (quote from [14] paraphrased by [15]). A thorough empirical and theoretical comparison of generative and discriminative classifiers is given in [15], in which the authors conclude that a generative classifier indeed has a higher asymptotic error when the number of training examples grows large. However, a generative classifier may approach its (higher) asymptotic error much faster than the discriminative model.

2.4 Previous Work

There has been much research into text classification, and the dominating approach has shifted from handcrafted rule-based methods to more recent automatically trained probabilistic and/or geometric methods [16]. An example of a classification rule in a rule-based system is shown in Figure 2.6.

if (wheat ∧ farm) ∨
(wheat ∧ commodity) ∨
(bushels ∧ export) ∨
(wheat ∧ agriculture) ∨
(wheat ∧ tonnes) ∨
(wheat ∧ winter ∧ ¬soft) **then** $\hat{c}$ = "wheat"

Figure 2.6: Example classification rule for the class wheat. Reproduced from [17].

The problem with this rule-based approach is not necessarily the resulting rules, but mainly the process of constructing the rules. The construction is time-consuming and often involves consulting a domain expert. The difference with later approaches is the shift from manual construction of the rules to manual construction of the classifier and learning algorithms, in addition to the manual labeling of training documents required by supervised learning. In this supervised learning paradigm, the manual work does not involve the inference itself, but rather the manual classification of training documents to be passed to an automated learning (inference) algorithm.

An overview over work regarding naive Bayes text classification is given in [9], including an evaluation of the two event models previously discussed in this chapter for different vocabulary sizes. In [10] different improvements to the standard naive Bayes model are discussed and evaluated.

In [18] the authors analyze why the classification accuracy for the naive Bayes classifier is surprisingly high in many cases, even though its posterior probability estimates are poor because of the conditional independence assumption.

2.4.1 Spoken Document Classification

The simplest way of constructing a spoken document classification system is concatenating an automatic speech recognizer with a text-based classification system. Unfortunately, the accuracy of current speech recognizers degrade significantly when faced with unconstrained conversational speech or varying acoustical environments [19]. Since the relative benefit of considering multiple hypotheses is expected to increase with decreasing recognition accuracy [20] [21], there has been much work going into the joint field of spoken document classification.

An insightful generalization emphasized in [22] is that the textual representation of a spoken document is probabilistic and can be seen as a distribution over its possible transcriptions. Hence, the features used for classification should be treated as proba-

bilistic quantities. Seen from this perspective, it is clearly suboptimal to consider the distribution over transcriptions $P(\mathbf{w}|\mathbf{o})$ only at its maximum as is done in 1-best classification. Additionally, when the uncertainty contained in the transcription distribution is large (high WER), the probability mass is distributed more evenly across hypotheses than when the uncertainty is smaller (low WER). With a small uncertainty, most of the probability mass is contained in the 1-best hypothesis, and there is not much to gain from considering the other unlikely hypotheses.

A spoken document classifier is proposed in [23]. They are using a multinomial naive Bayes classifier combined with expected word counts feature extraction. They employ χ^2 feature selection to select the classification vocabulary. Their results on the switchboard corpus show that using expected word counts is better than 1-best, both for feature (word) selection and for classification. They advise against feature selection from 1-best data by arguing that selecting a word because it discriminates well based on 1-best might not work well, since it could be the case that it cannot be reliably detected or it has a large insertion probability by the speech recognizer. The expected word counts are not computed directly from a recognition lattice, instead they are computed by using what they call a *HMM word spotter*, which is a special-purpose speech recognizer that aims to only recognize words chosen by the feature selector. They use a forward-backward algorithm to compute the posterior probabilities for the classification vocabulary words at all time steps. To calculate the expected word counts, instead of summing all the posteriors for each word, they only include a posterior in the sum when it reaches a local maximum on the time axis.

Another spoken document classifier is proposed in [24], including an update in [25] giving better results because of an improved ASR engine. In the first paper, the word accuracy in the recognition transcripts was 22%, and the word error rate was "well into the 90%'s". On these recognition transcripts they obtained a classification accuracy (for 10 topics) of 74% with a classification vocabulary of 4,600 words. They used a smoothed unigram language model for each topic (equivalent to naive Bayes except for the smoothing). They used the χ^2 feature selection criterion to reduce the vocabulary size. Both feature selection and classification were performed on 1-best recognition transcripts. The second paper left the classifier unchanged, only the ASR engine is improved to a word error rate "in the 40%'s". With this new recognizer, the same unigram language model classifier obtains an accuracy of 99.2%, the same as what they obtain on the reference transcripts.

There are also proposed several methods for segmenting speech into topics. Many of these are using a hidden Markov model in which each state represents a topic. Since a hidden Markov model is a generative model, the observation model for each state has

to be generative, and a topic-dependent language model is chosen in most approaches. Both [26] and [27] use a unigram language model equivalent to multinomial naive Bayes. In the latter paper a trigram language model is also tested, but unigram was found to be better.

As previously mentioned, spoken document classification is closely related to spoken document retrieval, in which a list of documents is returned in response to a given query. A lattice-based approach to spoken document retrieval is proposed in [21], where expected word counts are estimated from the lattice and subsequently used to produce a ranking of the documents. They report a significant relative improvement by using expected counts from lattices instead of 1-best word counts.

— 3 —

Methods & Implementation

We have implemented two spoken document classification systems by combining an automatic speech recognizer with the two classification algorithms multinomial naive Bayes and logistic regression. Our combination of expected word count feature extraction and the naive Bayes classifier is similar to the approach presented in [23]. But instead of their use of χ^2 feature selection, we used information gain feature selection, and in addition to naive Bayes, we also implemented logistic regression classification.

The systems are implemented in a single Java application. The application takes raw audio input and passes it to an automatic speech recognizer, from which the results are passed to the feature extractor and finally to the classifier. A flow chart of the process is shown in Figure 3.1.

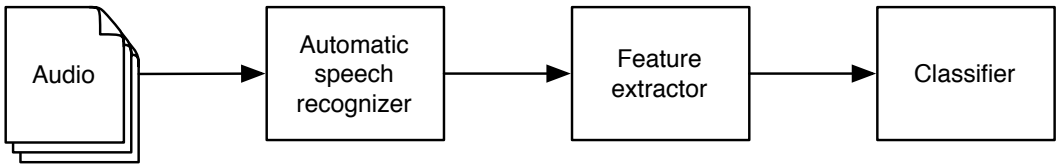


Figure 3.1: Flow chart of the classification system.

3.1 The Automatic Speech Recognizer

The automatic speech recognizer is based on the CMU Sphinx-4 open source framework for speech recognition [28], which is written in Java. Acoustic features in form of 13 mel-frequency cepstral coefficients (MFCCs) [29] are extracted from the audio signal, then normalized by their means to reduce convolutional channel distortion (cepstral mean normalization) [30], and finally their time deltas and double-deltas are also included to form the complete 39-element acoustic feature vector.

These acoustic feature vectors are then used in an HMM-based recognition system in which each HMM represents a cross-word context-dependent phone (triphone). The observation probability density for each HMM state is given by a mixture model consisting of eight Gaussians. For recognition, a grammar network is constructed by these HMMs according to a pronunciation lexicon and a trigram language model, and then a time-synchronous Viterbi beam search is performed on the grammar network to obtain the recognition hypotheses in form of a lattice.

Both the acoustic model and the language model used in this implementation are trained on the 1997 English Broadcast News Speech (HUB4) corpus[1] provided by the Linguistic Data Consortium. The vocabulary contains 64,000 words.

3.2 The Feature Extractor

The automatic speech recognizer returns a lattice representing multiple hypotheses with corresponding probabilities. The feature extractor converts a text string or a lattice to a D-dimensional feature vector that can be used directly for classification. We are using the bag-of-words representation, in which each of the D elements represents a word, and the corresponding value is the number of occurrences of that word. In order to save memory, a sparse vector representation is used for the feature vectors, in which only nonzero values are listed along with the corresponding word indices.

3.2.1 Word Normalization and Stemming

We start the feature extraction phase by normalization and stemming of the words. The normalization done on ASR output only consists of lowercasing all letters. However, the system should also be able to accept the reference transcripts, both for WER calculations and for reference classification. Because of this we have implemented some additional normalization which involves removal of punctuation, and conversion of numbers to words (eg. `2000` to `two thousand`).

Stemming is a procedure also known as suffix stripping, which reduces words to their root forms. Two examples of English stemming are `stemming` to `stem`, and `procedures` to `procedure`. The stemming procedure reduces the effective vocabulary size and can therefore shorten the feature vector. Since the exact form of a word is assumed to have little information about the topic label, stemming could also help improve accuracy. We are using the Porter stemming algorithm for English [31].

3.2.2 Counting Word Occurrences

We have implemented two different ways of counting word occurrences in the lattice. The first one counts the number of occurrences in the 1-best Viterbi path, while the other one counts the expected number of occurrences in the lattice. As seen in Chapter 2, computing the expected number of occurrences requires computation of the posterior probability for each node in the lattice, i.e. the probability that the node occurs on a random path through the lattice. Once the posterior computation is done, the expected

(1) http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC98S71

count for each word is obtained by summing the posteriors for all nodes corresponding to that word.

Let $\mathbb{V}$ be the set of all vertices (nodes) in the lattice, and $\mathbb{W}$ the set of all words in the vocabulary of the speech recognizer (and hence the vocabulary of the lattice). For a given vertex v, let $w(v)$ be the corresponding word. The computation of the expected number of occurrences of word w is then simply implemented by using the forward-backward algorithm as described in Chapter 2. The forward pass is given in Algorithm 1, the backward pass in Algorithm 2, and finally the expected word calculation is done in Algorithm 3.

Algorithm 1 Calculating forward variables.

$\alpha(v_S) := 1$
for all $v \in$ topologically_sorted$(\mathbb{V} \setminus \{v_S\})$ **do**
    $\alpha(v) := 0$
    for all $e \in$ incoming_edges(v) **do**
        $\alpha(v) := \alpha(v) + \alpha($source_vertex$(e)) * $ edge_probability(e)
    end for
end for

Algorithm 2 Calculating backward variables.

$\beta(v_E) := 1$
for all $v \in$ reverse_topologically_sorted$(\mathbb{V} \setminus \{v_E\})$ **do**
    $\beta(v) := 0$
    for all $e \in$ outgoing_edges(v) **do**
        $\beta(v) := \beta(v) + \beta($destination_vertex$(e)) * $ edge_probability(e)
    end for
end for

Algorithm 3 Calculating expected word counts.

for all $w \in \mathbb{W}$ **do**
    $c(w) := 0$
end for
for all $v \in \mathbb{V}$ **do**
    $c(w(v)) := c(w(v)) + \alpha(v) * \beta(v)/\alpha(v_E)$
end for

3.2.3 *Word Selection*

The number of distinct words in the lattice is upper-bounded by the number of words in the vocabulary used by the speech recognizer, which in our case is 64,000 words. We tried both χ^2 and information gain feature selection, and in practice information gain gave marginally better results, so we chose to go with information gain feature selection in the final implementation. In order to be able to compute the information gain, the continuous values that arise from the expected value computation are discretized by recursive splitting according to an information gain criterion, as proposed in [8] and explained in Section 2.2.2.

3.3 The Naive Bayes Classifier

The open source Weka data mining software [32] (written in Java) was used for implementation of the multinomial naive Bayes classifier. The classifier is trained with Laplace smoothing (add-one smoothing) for both the class-conditional parameters and the class priors.

When a trained classifier is passed a test instance $\mathbf{x}$, it returns a vector containing the posterior distribution over classes, which is obtained by normalization of the classifier output $P(\mathbf{x}|c)P(c)$ over all classes:

$$P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c)P(c)}{\sum_{c \in \mathbb{C}} P(\mathbf{x}|c)P(c)}.$$

3.4 The Logistic Regression Classifier

The LibLINEAR library [33] was used for implementation of the logistic regression classifier. The LibLINEAR library applies a trust region Newton method to efficiently maximize the log-likelihood during training of the logistic regression classifier [34]. An L_2 regularization term is subtracted from the log likelihood in order to get an objective function that penalizes large weights, since this can prevent overfitting [35].

We used a Java wrapper for LibLINEAR in order to call it from our Java application. The LibLINEAR logistic regression classifier also returns a vector containing the posterior distribution over classes when given a test instance, from which we can pick the maximum as the MAP estimate.

— 4 —
Results & Discussion

In this chapter, we describe our experiments and report the results obtained with the implementation described in the previous chapter.

4.1 Data Set

The experiments were performed on news stories from the English part of the Topic Detection and Tracking data from 2003 (TDT4), provided by the Linguistic Data Consortium[1]. The English part of the data consists of 874 categorized news stories from six different broadcasters. The distribution over categories and broadcasters is shown in Table 4.4. The broadcaster names behind the acronyms and the names of the news shows are listed in Appendix A. The average news story length is 250 words.

Category	ABC	CNN	MNB	NBC	PRI	VOA	Total
Elections	12	21	6	16	31	45	131
Scandals/Hearings	9	10	1	5	8	13	46
Legal/Criminal	8	20	0	10	13	13	64
Natural Disasters	12	22	2	8	5	23	72
Accidents	9	20	4	9	11	30	83
Violence or War	30	54	9	22	28	57	200
Science	2	4	0	1	5	0	12
New Laws	1	5	0	2	2	12	22
Sports	2	23	0	0	1	10	36
Political Meetings	6	20	2	7	21	45	101
Celebrities	10	20	0	6	9	15	60
Miscellaneous	3	6	0	1	13	24	47
Total	104	225	24	87	147	287	874

Table 4.1: Distribution over categories and broadcasters in the TDT4 data set.

4.2 Automatic Speech Recognition Results

We have run two recognition rounds with different pruning beam widths in order to evaluate the systems on recognition data with different word error rates.

(1) http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2005S11

As can be seen from Table 4.2, the overall WER of the stemmed 1-best output is 49.8% for the whole data set when we set a relatively generous pruning beam width. The error rates vary a bit across the broadcasters, with the two radio broadcasters PRI (Public Radio International) and VOA (Voice of America) being the ones with the lowest WERs at 51.0% and 44.8% respectively. The television broadcasters often have more background noise than radio, and this could be the reason why their WERs are a little higher. Stemming gives an absolute decrease in the WER of 1.5%, and this improvement comes mainly from a reduced number of substitutions caused by recognition of the wrong form of a word.

As can be seen from Table 4.3, the overall WER of the stemmed 1-best output with more aggressive pruning is 84.5% for the whole data set. The absolute reduction in WER after stemming is 0.8%, also mainly from a reduced number of substitutions.

	ABC	CNN	MNB	NBC	PRI	VOA	Total
Unstemmed WER	52.71%	55.84%	56.45%	56.10%	52.49%	46.00%	51.30%
Stemmed WER	51.29%	54.25%	54.88%	54.46%	51.14%	44.39%	49.78%

Table 4.2: Recognition errors for the different broadcasters with less pruning.

	ABC	CNN	MNB	NBC	PRI	VOA	Total
Unstemmed WER	85.25%	87.64%	87.13%	87.28%	85.52%	83.29%	85.26%
Stemmed WER	84.43%	86.80%	86.34%	86.49%	84.68%	82.51%	84.45%

Table 4.3: Recognition errors for the different broadcasters with more pruning.

4.3 Classification Evaluation Methodology

We evaluate the naive Bayes and logistic regression classifiers with both the low WER and high WER recognition data, giving a total of four different combinations or "scenarios". We report the stratified five-fold cross-validation accuracy obtained in each scenario. The results are compared to the results obtained by using the same classifiers and the same cross-validation method on the official TDT4 reference transcripts, which should be an upper-bound on the achievable accuracy.

A range of different classification vocabulary sizes is tested in each scenario. The *ASR vocabulary* size is 64,000 in all cases, but the *classification vocabulary* size is reduced

	Less pruning		More pruning	
Category	Unstemmed	Stemmed	Unstemmed	Stemmed
Elections	55.06%	53.94%	86.24%	85.48%
Scandals/Hearings	52.99%	51.78%	86.44%	85.63%
Legal/Criminal	50.65%	49.16%	85.14%	84.30%
Natural Disasters	58.32%	57.19%	87.42%	86.71%
Accidents	51.15%	49.88%	85.44%	84.73%
Violence or War	49.03%	46.58%	83.97%	83.05%
Science	51.20%	49.88%	84.32%	83.43%
New Laws	39.05%	36.94%	80.97%	79.99%
Sports	56.64%	55.71%	86.46%	85.86%
Political Meetings	48.37%	46.88%	84.73%	83.90%
Celebrities	48.35%	47.46%	85.15%	84.23%
Miscellaneous	46.79%	45.13%	84.82%	84.08%
Total	51.30%	49.78%	85.26%	84.45%

Table 4.4: Recognition errors for the different topics.

from this upper bound by using information gain feature selection. It should be noted that in practice the upper bound is the number of distinct words occurring in the recognition data. There are around 20,000 distinct words occurring in the recognition lattices, around 17,000 in the 1-best hypotheses, and around 9,600 in the reference transcriptions.[2]

Each fold in the stratified cross-validation is performed by first ranking the words (features) by their information gain with respect to the training data for that fold. For each classification vocabulary size k to be tested, the top-k words from this ranking are used to train the classifier on the same training data, before testing is performed on the testing data for the fold. The reported accuracy is the average test accuracy obtained in this manner across all five folds for a given vocabulary size.

4.4 Classification Evaluation Results

4.4.1 Feautere Selection

Since the feature selection is done independently in each fold (on the training portion), the selected features will vary across folds. However, in Table 4.5 we have listed the top ranked words with respect to the information gain on the whole data set to give an idea about the kind of words that are selected.

(2) Both the lattice and Viterbi numbers are averages across the higher and lower WER data. In fact, the number of distinct words is a little larger with higher WER than with lower WER.

Rank	Word	Information gain
1	presid	0.2536
2	yemen	0.2511
3	coal	0.2278
4	earthquak	0.2270
5	terrorist	0.2227
6	attack	0.2216
7	sailor	0.2185
8	elect	0.2095
9	clinton	0.1923
10	prison	0.1830

Table 4.5: The top 10 ranked words by information gain on the expected word counts from the 50% WER recognition data. Notice the stemming from `president(s)` *to* `presid`*,* `earthquake(s)` *to* `earthquak` *and* `election(s)` *to* `elect`*.*

4.4.2 *Naive Bayes*

The results for the naive Bayes classifier for lower and higher WER are shown in Figure 4.1.

The results for lower WER are shown in Figure 4.1a. We see that there is almost no loss in classification accuracy with 1-best word counts, compared to the reference, for vocabulary sizes up to around 200 words. The expected word counts give a slightly lower accuracy than the 1-best word counts for these small vocabulary sizes. For larger vocabulary sizes, 1-best word counts and expected word counts have approximately the same accuracy, with increasing loss compared to the reference accuracy. Both the 1-best and expected word count classifiers reach a maximum accuracy of approximately 88% with a 700-word vocabulary, and both start losing accuracy when more words are added to the vocabulary. The same pattern is seen for the reference, but its maximum accuracy of 91% is reached at a larger vocabulary size of 980 words.

The results for higher WER are shown in Figure 4.1b. The reference results are naturally the same as in Figure 4.1a. The large number of recognition errors causes a significant loss in accuracy for classification with both the 1-best and expected word counts, but the expected word counts give a noticeably higher accuracy than the 1-best word counts for all vocabulary sizes larger than around 15 words. Classification with expected word counts reaches a maximum accuracy of 50% at a vocabulary size of 950 words, while classification with 1-best word counts reaches its maximum of 46% accuracy at a vocabulary size of 230 words.

4.4.3 *Logistic Regression*

The results for the logistic regression classifier with lower and higher WER are shown in Figure 4.2.

In Figure 4.2a, the results for lower WER are shown. The accuracy with both 1-best and expected word counts increase with increasing vocabulary size up to 200 words, but for larger vocabulary sizes the classification accuracy is almost constant at 90% with a slight increase to 92% for the 1-best word counts with vocabulary sizes above 1,000 words. The logistic regression reference classifier obtains a maximum accuracy of 96% at a vocabulary size of 490 words.

The results for higher WER are shown in Figure 4.2b. The reference results are still from the manual transcriptions and thus have the same results as in Figure 4.2a. As with naive Bayes, the large number of recognition errors causes a significant loss in accuracy for classification with both the 1-best and expected word counts, but the expected word counts give a noticeably higher accuracy than the 1-best word counts for all vocabulary sizes larger than around 15 words. With this high WER, logistic regression performs best when including all the words (no feature selection), giving a 57% accuracy for expected word counts and 53% accuracy for 1-best word counts.

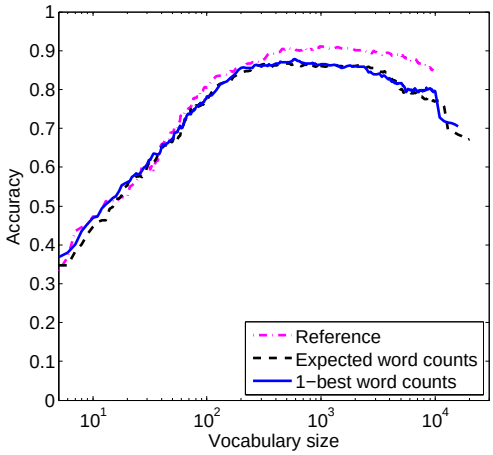
A confusion matrix for the logistic regression classifier is shown in Table 4.6, and a confusion matrix for the naive Bayes classifier is shown in Table 4.7.

4.4.4 *Comparing Logistic Regression and Naive Bayes*

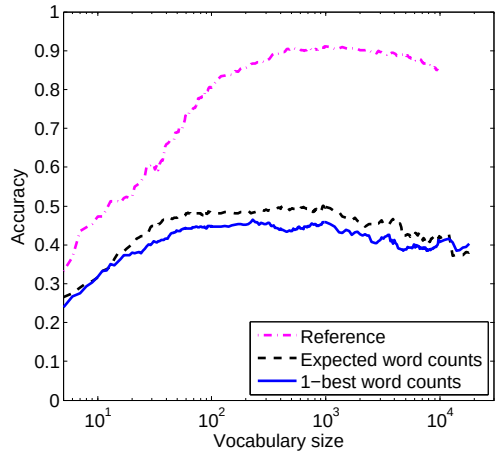
Logistic regression and naive Bayes are compared in Figure 4.3 for 1-best word counts and in Figure 4.4 for expected word counts.

On the 55% WER recognition data, the logistic regression classifier consistently outperforms the naive Bayes classifier across all vocabulary sizes.

On the 85% WER recognition data, logistic regression and naive Bayes have almost the same accuracies for all vocabularies under 1,000 words. While logistic regression gains accuracy when adding more words to the vocabulary after this point, naive Bayes instead loses accuracy.

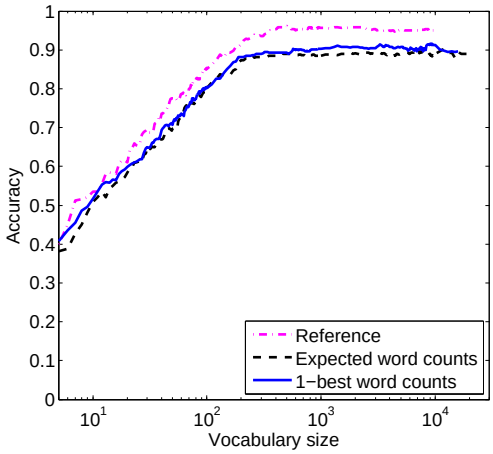


(a) 50% word error rate.

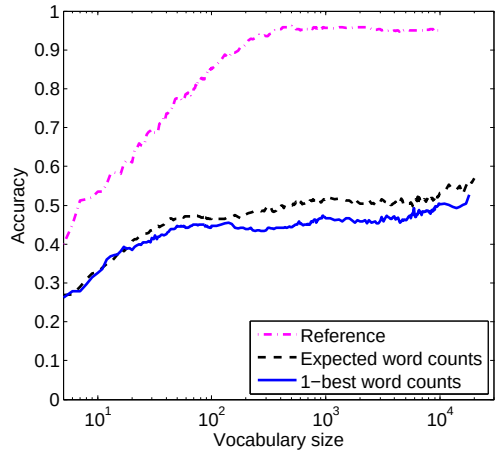


(b) 85% word error rate.

Figure 4.1: Five-fold cross-validation accuracy for the naive Bayes classifier.

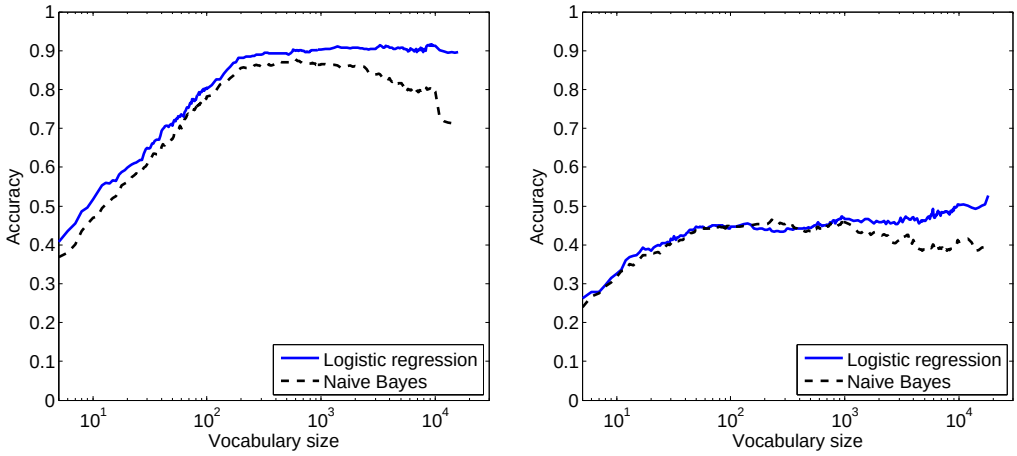


(a) 50% word error rate.



(b) 85% word error rate.

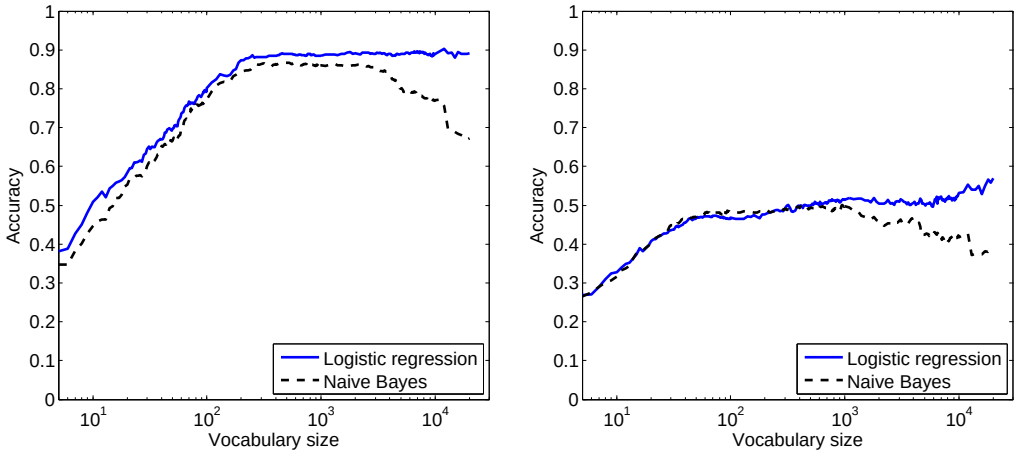
Figure 4.2: Five-fold cross-validation accuracy for the logistic regression classifier.



(a) 50% word error rate.

(b) 85% word error rate.

Figure 4.3: Five-fold cross-validation accuracy for 1-best word counts.



(a) 50% word error rate.

(b) 85% word error rate.

Figure 4.4: Five-fold cross-validation accuracy for expected word counts.

Classified as:	a	b	c	d	e	f	g	h	i	j	k	l
a = Elections	**126**	1	0	0	1	2	0	0	0	1	0	0
b = Scandals/Hearings	1	**42**	1	0	0	1	0	0	0	1	0	0
c = Legal/Criminal	0	0	**64**	0	0	0	0	0	0	0	0	0
d = Natural Disasters	0	0	0	**70**	2	0	0	0	0	0	0	0
e = Accidents	0	0	1	4	**78**	0	0	0	0	0	0	0
f = Violence or War	20	0	3	2	1	**167**	0	0	0	4	1	2
g = Science	0	0	0	3	0	0	**9**	0	0	0	0	0
h = New Laws	2	0	0	1	0	1	0	**15**	0	3	0	0
i = Sports	0	1	0	1	0	8	0	0	**26**	0	0	0
j = Political Meetings	2	1	1	0	1	2	0	0	0	**93**	1	0
k = Celebrities	6	1	0	0	1	0	0	1	0	5	**46**	0
l = Miscellaneous	14	0	0	1	0	4	0	0	0	3	1	**24**

Table 4.6: Cross-validation confusion matrix for the naive Bayes classifier with expected word counts on the 50% WER data and a classification vocabulary of 750 words.

Classified as:	a	b	c	d	e	f	g	h	i	j	k	l
a = Elections	**121**	0	0	0	0	7	0	0	0	0	1	2
b = Scandals/Hearings	1	**40**	1	0	0	0	0	0	2	2	0	0
c = Legal/Criminal	0	0	**64**	0	0	0	0	0	0	0	0	0
d = Natural Disasters	0	0	0	**68**	2	0	0	0	1	1	0	0
e = Accidents	0	0	0	3	**78**	0	0	1	1	0	0	0
f = Violence or War	4	0	1	0	1	**185**	0	0	2	4	1	2
g = Science	0	0	0	1	0	1	**10**	0	0	0	0	0
h = New Laws	0	0	0	0	0	3	0	**16**	0	3	0	0
i = Sports	0	1	0	0	0	1	0	0	**34**	0	0	0
j = Political Meetings	1	2	0	0	0	3	0	0	1	**93**	0	1
k = Celebrities	5	0	0	1	0	1	0	0	1	2	**50**	0
l = Miscellaneous	6	0	0	1	2	3	0	0	1	4	0	**30**

Table 4.7: Cross-validation confusion matrix for the logistic regression classifier with expected word counts on the 50% WER data and a classification vocabulary of 750 words.

4.5 Discussion

We can observe some interesting patterns in the reported results. First, the logistic regression classifier outperforms or performs as well as the naive Bayes classifier in almost all of the tests. Their respective maxima are 92% versus 88% on the 50% WER data, and 53% versus 46% on the 85% WER data.

Another apparent difference is that while the naive Bayes classifier reaches a clear maximum before getting less accurate for bigger vocabulary sizes, the accuracy for the logistic regression classifier is almost a nondecreasing function with respect to the vocabulary size. On the 50% WER recognition data, neither of the classifiers gain any benefit from the the words ranked below position 600-800 by the information gain criterion. However, it is apparent that the logistic regression classifier is able to ignore these irrelevant or noisy words, while the naive Bayes classifier instead is negatively impacted.

On the 50% WER recognition data, there seems to be no advantage of using the expected word counts over the 1-best word counts. For the logistic regression classifier, 1-best word counts actually give slightly better results than expected word counts. However, when the automatic speech recognizer has a WER of 85%, usage of expected word counts gives a consistent increase in accuracy over simple 1-best word counts. The absolute increase in accuracy by using expected word counts is 4% for both naive Bayes and logistic regression. The results indicate that the advantage of using expected word counts increases with increasing speech recognition error rates, which is consistent with the literature. An intuitive explanation for this is that more of the probability mass in the transcription distribution is contained in hypotheses other than the 1-best, and therefore are 1-best word counts a less accurate estimate of the true transcription distribution.

The 92% classification accuracy that was obtained with the logistic regression classifier is encouraging considering the relatively high word error rate of 50%. With a very high word error rate of 85%, the classifier is able to make the correct prediction half of the time.

— 5 —

Conclusion & Future Work

We have implemented two systems for spoken document classification by combining an automatic speech recognizer with the two classification algorithms naive Bayes and logistic regression. The focus has been on how to deal with the inherent uncertainty in the output of the speech recognizer. We performed feature extraction by computing expected word counts from speech recognition lattices by using a forward-backward algorithm, and words were selected for the classification vocabulary according to the information gain feature selection criterion. The systems were evaluated by performing stratified cross-validation on broadcast news stories, and the classification accuracy is measured with different configurations and on recognition output with different word error rates. The results show that a relatively high classification accuracy can be obtained with word error rates around 50%, and that the benefit of extracting features from lattices instead of 1-best transcripts increases with increasing word error rates.

To further improve the classification accuracy for this data set, the most apparent route is to try and improve the accuracy of the speech recognizer. Other classifier algorithms could also be explored. On a more general note, if we are faced with a situation of conversational speech with varying acoustic environment where the WER is high and hard to reduce, there is a significant benefit of using expected word counts instead of 1-best word counts.

In this thesis, we have assumed that the boundaries of the news stories are known. In many real-world situations however, this is not the case, and segmentation of audio by topic is an interesting problem for further research.

— A —

The TDT4 Broadcasters and News Shows

Acronym	Broadcaster	News show
ABC	American Broadcasting Company	"World News Tonight"
CNN	Cable News Network	"Headline News"
MNB	Microsoft/National Broadcasting Company	"News with Brian Williams"
NBC	National Broadcasting Company	"NBC Nightly News"
PRI	Public Radio International	"The World"
VOA	Voice of America	English news programs

Table A.1: The English news shows in the TDT4 data set.

References

[1] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, February 1989.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[3] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, May 2001.

[4] Janne Pylkkönen. New pruning criteria for efficient decoding. In *INTERSPEECH'05*, pages 581–584, 2005.

[5] Yen-Lu Chow and Richard Schwartz. The N-Best algorithm: an efficient procedure for finding top N sentence hypotheses. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 199–202, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics.

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[7] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[8] Usama M Fayyad and Keki B Irani. *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning*, volume 2, pages 1022–1027. Morgan Kaufmann, 1993.

[9] Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*, pages 41–48, 1998.

[10] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.

[11] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[13] Takayoshi Yoshimura, Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Tadashi Kitamura. Simultaneous Modeling Of Spectrum, Pitch And Duration In HMM-Based Speech Synthesis. *Proceedings of Eurospeech 1999*, 1999.

[14] Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, September 1998.

[15] Andrew Y. Ng and Michael I. Jordan. On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes, 2002.

[16] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[17] Chidanand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251, July 1994.

[18] David J. Hand and Keming Yu. Idiot's Bayes—Not So Stupid After All? *International Statistical Review*, 69(3):385–398, 2001.

[19] John Garofolo Jonathan Fiscus, Jerome Ajot. The rich transcription 2007 meeting recognition evaluation. In *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 373–389. Springer Berlin / Heidelberg, 2008.

[20] Jonathan Mamou, David Carmel, and Ron Hoory. Spoken document retrieval from call-center conversations. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR 2006, pages 51–58, New York, NY, USA, 2006. ACM.

[21] Ciprian Chelba, Jorge Silva, and Alex Acero. Soft indexing of speech content for search in spoken documents. *Comput. Speech Lang.*, 21:458–478, July 2007.

[22] Matthew A. Siegler. *Integration of continuous speech recognition and information retrieval for mutually optimal performance*. PhD thesis, Pittsburgh, PA, USA, 1999. AAI9964617.

[23] J. McDonough, K. Ng, P. Jeanrenaud, H. Gish, and J.R. Rohlicek. Approaches to topic identification on the switchboard corpus. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume i, pages I/385 –I/388 vol.1, April 1994.

[24] L. Gillick, J. Baker, J. Baker, J. Bridle, M. Hunt, Y. Ito, S. Lowe, J. Orloff, B. Peskin, R. Roth, and F. Scattone. Application of large vocabulary continuous speech recognition to topic and speaker identification using telephone speech. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 471 –474 vol.2, April 1993.

[25] B. Peskin, S. Connolly, L. Gillick, S. Lowe, D. McAllaster, and V. Nagesha. Improvements in switchboard recognition and topic identification. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 303 –306 vol. 1, May 1996.

[26] J.P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt. A hidden Markov model approach to text segmentation and event tracking. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 333 –336 vol.1, May 1998.

[27] M. Sherman and Yang Liu. Using hidden Markov models for topic segmentation of meeting transcripts. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 185 –188, December 2008.

[28] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Technical report, Mountain View, CA, USA, 2004.

[29] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357 – 366, August 1980.

[30] Jr. Jankowski, C.R., H.-D.H. Vo, and R.P. Lippmann. A comparison of signal processing front ends for automatic word recognition. *Speech and Audio Processing, IEEE Transactions on*, 3(4):286 –293, July 1995.

[31] M. F. Porter. An algorithm for suffix stripping. In Karen Sparck Jones and Peter Willett, editors, *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[32] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[33] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LibLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[34] Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 561–568, New York, NY, USA, 2007. ACM.

[35] Andrew Y. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.