Phoneme-Based Topic Spotting on the Switchboard Corpus

M. W. Theunissen

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Science in Electronic Engineering
at the
University of Stellenbosch

Study Leader: **Prof. J. A. du Preez**

March 2002

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

March 2002

Abstract

The field of topic spotting in conversational speech deals with the problem of identifying "interesting" conversations or speech extracts contained within large volumes of speech data. Typical applications where the technology can be found include the surveillance and screening of messages before referring to human operators. Closely related methods can also be used for data-mining of multimedia databases, literature searches, language identification, call routing and message prioritisation.

The first topic spotting systems used words as the most basic units. However, because of the poor performance of speech recognisers, a large amount of topic-specific hand-transcribed training data is needed. It is for this reason that researchers started concentrating on methods using phonemes instead, because the errors then occur on smaller, and therefore less important, units. Phoneme-based methods consequently make it feasible to use computer generated transcriptions as training data.

Building on word-based methods, a number of phoneme-based systems have emerged. The two most promising ones are the Euclidean Nearest Wrong Neighbours (ENWN) algorithm and the newly developed Stochastic Method for the Automatic Recognition of Topics (SMART). Previous experiments on the Oregon Graduate Institute of Science and Technology's Multi-Language Telephone Speech Corpus suggested that SMART yields a large improvement over ENWN which outperformed competing phoneme-based systems in evaluations. However, the small amount of data available for these experiments meant that more rigorous testing was required.

In this research, the algorithms were therefore re-implemented to run on the much larger Switchboard Corpus. Subsequently, a substantial improvement of SMART over ENWN was observed, confirming the result that was previously obtained. In addition to this, an investigation was conducted into the improvement of SMART. This resulted in a new counting strategy with a corresponding improvement in performance.

Opsomming

Die veld van onderwerp-herkenning in spraak het te doen met die probleem om "interessante" gesprekke of spraaksegmente te identifiseer tussen groot hoeveelhede spraakdata. Die tegnologie word tipies gebruik om gesprekke te verwerk voor dit verwys word na menslike operateurs. Verwante metodes kan ook gebruik word vir die ontginning van data in multimedia databasisse, literatuur-soektogte, taal-herkenning, oproep-kanalisering en boodskap-prioritisering.

Die eerste onderwerp-herkenners was woordgebaseerd, maar as gevolg van die swak resultate wat behaal word met spraak-herkenners, is groot hoeveelhede hand-getranskribeerde data nodig om sulke stelsels af te rig. Dit is om hierdie rede dat navorsers tans foneemgebaseerde benaderings verkies, aangesien die foute op kleiner, en dus minder belangrike, eenhede voorkom. Foneemgebaseerde metodes maak dit dus moontlik om rekenaargegenereerde transkripsies as afrigdata te gebruik.

Verskeie foneemgebaseerde stelsels het verskyn deur voort te bou op woordgebaseerde metodes. Die twee belowendste stelsels is die "Euclidean Nearest Wrong Neighbours" (ENWN) algoritme en die nuwe "Stochastic Method for the Automatic Recognition of Topics" (SMART). Vorige eksperimente op die "Oregon Graduate Institute of Science and Technology's Multi-Language Telephone Speech Corpus" het daarop gedui dat die SMART algoritme beter vaar as die ENWN-stelsel wat ander foneemgebaseerde algoritmes geklop het. Die feit dat daar te min data beskikbaar was tydens die eksperimente het daarop gedui dat strenger toetse nodig was.

Gedurende hierdie navorsing is die algoritmes dus herimplementeer sodat eksperimente op die "Switchboard Corpus" uitgevoer kon word. Daar is vervolgens waargeneem dat SMART aansienlik beter resultate lewer as ENWN en dit het dus die geldigheid van die vorige resultate bevestig. Ter aanvulling hiervan, is 'n ondersoek geloods om SMART te probeer verbeter. Dit het tot 'n nuwe telling-strategie gelei met 'n meegaande verbetering in resultate.

Acknowledgements

I would like to thank the following for their help, support and encouragement:

- My study leader, Prof. J. A. du Preez, for his patience and advice.
- Konrad Scheffler, for working with me on the Eurospeech paper.
- Roland Kuhn, without whose help this entire project would have been impossible.
- Stefan Harbeck, for informing me which subset of the Switchboard Corpus had to be used.
- Auke Slotegraaf, for proofreading this thesis.
- Dirko van Schalkwyk, for assisting me with Linux related problems.
- My family, for all their love and support.
- Eanette, for her boundless patience and love during this long project.
- Marius and Rosemary Lategan, for all their words of encouragement.
- The Centre of Excellence in ATM and Broadband Networks and their Applications, for their financial support.
- And lastly, Carl Sagan, for writing the books which served as a source of inspiration during my studies. No one has ever succeeded in conveying the wonder, excitement and joy of science as widely as he did.

Contents

List of Figures

List of Tables

Acronyms

ASR	Automatic speech recognition
CE	Cross-entropy
CRIM	Centre de Recherche Informatique de Montréal
EM	Expectation maximisation
ENWN	Euclidean Nearest Wrong Neighbours
HMM(s)	Hidden Markov model(s)
ICSI	International Computer Science Institute
LPCC(s)	Linear predictive cepstral coefficient(s)
LVCSR	Large vocabulary continuous speech recognition
MAP	Maximum *a posteriori*
MLTS	Multi-Language Telephone Speech
OGI	Oregon Graduate Institute of Science and Technology
pdf(s)	Probability density function(s)
ROC	Receiver operating characteristic
SMART	Stochastic Method for the Automatic Recognition of Topics

Mathematical Notation

$\mathbf{C}$	Conversation vector
C_i	The ith element of $\mathbf{C}$
$\mathbf{T}$	Topic vector
T_i	The ith element of $\mathbf{T}$
$\mathbf{R}(\mathbf{C})$	Correct (right) topic vector corresponding to $\mathbf{C}$
R_i	The ith element of $\mathbf{R}(\mathbf{C})$
$\mathbf{W}(\mathbf{C})$	Nearest wrong topic vector to $\mathbf{C}$
W_i	The ith element of $\mathbf{W}(\mathbf{C})$
$E(\mathbf{C}, \mathbf{R}(\mathbf{C}), \mathbf{W}(\mathbf{C}))$	Error function
E_T	Total error
$\mathbf{x}$	Uncorrupted keystring
x_i	The ith phoneme of $\mathbf{x}$
$\mathbf{y}$	Corrupted keystring
y_i	The ith phoneme of $\mathbf{y}$
$P(\cdot)$	Probability
$Conf(a, b)$	Entry in confusion matrix corresponding to row a and column b
q_k	The kth phoneme in the phonemic alphabet
$E(Cnt(\mathbf{x}))$	Expected value of the "true" count of $\mathbf{x}$
$F(\mathbf{x})$	Occurrence frequency of $\mathbf{x}$
$\mathcal{C}$	Conversation
$\mathcal{T}$	Topic
l_s	Minkowski metric of order s

$\mathbf{v}$	Multi-dimensional feature vector
v_i	The ith element of $\mathbf{v}$
$\mathbf{z}$	Multi-dimensional stream of acoustic feature vectors
$\mathfrak{z}_i$	Acoustic feature vectors on which y_i is based
$f(\cdot)$	Probability density function
$H(z)$	Filter transfer function in the z-domain
I	Improvement in total training score during training of a phoneme-HMM
R	Overall recognition accuracy of the phoneme recognition front end
R_k	Front end's recognition accuracy for phoneme q_k
β	Binomial distribution
α	McNemar rejection threshold
P_2	Two-tailed McNemar probability

We know very little, and yet it is astonishing that we know so much, and still more astonishing that so little knowledge can give us so much power.

— Bertrand Russel

Chapter 1

Introduction

The NSA [United States National Security Agency] patent, granted on 10 August [1999], is for a system of automatic topic spotting and labelling of data. The patent officially confirms for the first time that the NSA has been working on ways of automatically analysing human speech. The NSA's invention is intended to automatically sift through human speech transcripts in any language. The patent document specifically mentions "machine-transcribed speech" as a potential source.

Bruce Schneier, author of Applied Cryptography, a textbook on the science of keeping information secret, believes the NSA currently has the ability to use computers to transcribe voice conversations. 'One of the holy grails of the NSA is the ability to automatically search through voice traffic. They would have expended considerable effort on this capability, and this indicates it has been fruitful,' he said.

— The Independent

1.1 Overview of Conversational Topic Spotting

The field of topic spotting in conversational speech deals with the problem of identifying "interesting" conversations or speech extracts contained within large volumes of speech data. Typical applications where the technology can be found include the surveillance and screening of messages before referring to human operators. Closely related methods

can also be used for data-mining of multimedia databases, literature searches, language identification, call routing and message prioritisation.

One of the major problems encountered when doing topic spotting in conversational speech is the imperfect transcriptions produced by automatic speech recognition (ASR) systems. In addition to this, human speech often covers topics that are never actually spoken by name. Both of these factors contribute significantly towards the difficulty of using computers when determining the topic(s) of a conversation[1]. However, various pattern recognition techniques exist that can be used to overcome these problems.

Retrieval is usually done by monitoring the occurrences of words or sub-word segments (e.g. phoneme[2] strings). Central to the idea of topic spotting is the concept of "usefulness". In order for a feature (e.g. a word or phoneme string) to be useful, it must occur a sufficient number of times so that reliable statistics can be gathered. A significant difference must also exist in the distribution of the specific feature in the wanted and unwanted data. The choice of which feature to use is important, since it will ultimately determine what the system is sensitive to. For example, a system based on phonemes may be sensitive to regional accents, while a word-based system is likely to be more sensitive to message content. The exact details of the application will dictate which feature is more useful.

1.1.1 Word-Based Methods

The first topic spotting systems [2, 3] used words as the most basic units. This implies sending a conversation through a speech recogniser that transforms it into words which are then used by the rest of the system. Existing techniques are mainly based on methods using language modelling [4, 5, 6, 7, 8, 9] or keyword spotting [2, 3, 10, 11, 12, 13, 14, 15, 16]:

[1] **In this thesis, a conversation refers to a single speech signal containing the speech of one or more individuals.**

[2] The phonemes of a language comprise a minimal theoretical set of units that are sufficient to convey all meaning in the language. A phoneme thus represents a single sound, playing the same role in conversational speech as a letter does in text. However, the actual sound produced when pronouncing a phoneme is called a phone. For more information refer to Deller *et al.* [1], pp. 115–116.

- **Language modelling**: Statistical models (usually one model for each topic) are created of the co-occurrence frequencies of keywords in the particular topics of interest. These topic models are subsequently used to determine the conversation's topic(s).

- **Keyword spotting**: The occurrences of only a *few* keywords are monitored during the topic spotting process. An information measure is then employed to determine how strongly the occurrence of each keyword indicates the presence of a topic. Selecting the keywords is very important. By allowing the user to just merely specify them is counter-intuitive. As a result, a number of sophisticated statistical techniques [10, 11] have been developed to determine which keywords to use.

The relatively poor performance of large vocabulary continuous speech recognition (LVCSR) systems hampers these approaches, since the generated word transcriptions are not of sufficient quality to be used during training of these topic spotters. As a result, a large amount of topic-specific hand-transcribed training data is needed, a situation that causes problems for practical applications.

1.1.2 Phoneme-Based Methods

To address the above mentioned problem, one can rather work with phonemes as the most basic units. A number of advantages of this approach are pointed out in [17], most notably the increased robustness to recognition errors during the topic spotting process. This is because the errors occur on smaller, and therefore less important, units. Other advantages include:

- A smaller set of units has to be recognised.

- Word boundaries, which are often not audible in speech, do not have to be determined.

- Information at the sub-word level can be incorporated into the topic spotting process.

As a result, phoneme-based methods make it feasible to use topic-specific computer generated phonemic transcriptions as training data. The main source of difficulty with this approach is the inaccuracy introduced by the phoneme recognition front end. Typical recognition accuracy ranges between 40 and 60 percent on speech data encountered over broadcast- and telephone-channels [18]. However, various methods based on dynamic programming [19] or hidden Markov models (HMMs) [20, 21, 22, 23] exist that can be used to compensate for insertion-, deletion- and substitution-errors which are introduced by the phoneme recogniser.

Existing systems are based on methods using language modelling [24, 25, 26, 27, 28, 29, 30] or keystring spotting [17, 31], where the concept of a keyword is generalised to that of a phoneme string (keystring).

1.1.3 System Diagram

A diagrammatic representation of a general topic spotting system is depicted in Figure 1.1.

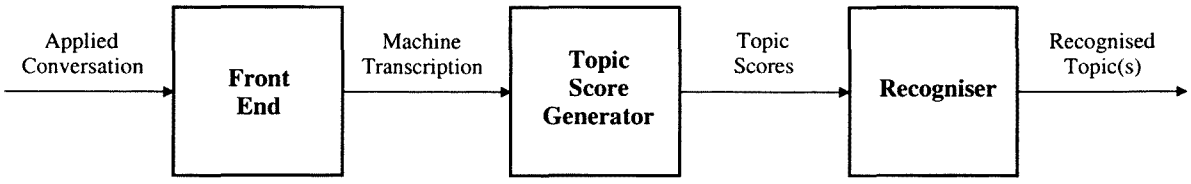


Figure 1.1: *Diagrammatic representation of a general topic spotting system.*

A description of each component follows below.

1.1.3.1 Front End

The function of the front end is to convert a raw speech message, presented at its input, into a more acceptable format (e.g. words or phonemes) that can be used by the rest of the topic spotting system. If words are chosen as the most basic units, select a LVCSR

system or a word spotter. However, if the most basic units are phonemes, use a phoneme recogniser instead.

Various factors influence the performance of these ASR systems, most notably the quality of the speech data that has to be processed. In particular, the speech data used during topic spotting is usually of low quality and as a result possesses the following complicating characteristics:

- since it is received via broadcast- or telephone-channels it suffers from microphone- and communication channel-distortion,
- varying background noise,
- speaker variability due to stress, emotion and the Lombard effect,
- changes in accent/language,
- speakers are directing their communication at other humans, and not making any special effort towards clear articulation,
- speech is continuous, with the words not clearly separated, and
- the speech data is of unlimited vocabulary and context. Conversations can thus contain unknown words and language patterns.

It is for these reasons that the areas of man-machine interaction and voice telephony in adverse environments have emerged as a major research problem. However, ASR is fast becoming a mature technology, with great advances being made in recent years. It is therefore hoped that the current poor performance of these systems will greatly be improved in the not-too-distant future.

The inability of current speech recognition systems to generate good transcriptions does not imply that topic spotting is impossible. For example, consider the situation when one is listening to a conversation. In order to determine the topic(s), it is not necessary to

recognise each word or sound correctly. In fact, it is usually even possible to determine the topic(s) when the conversation is conducted in a language with which one is only partially familiar. Reliable topic spotting in conversational speech should thus be possible in spite of the poor performance of ASR systems.

To transcribe an applied conversation, the front end first extracts acoustic feature vectors. Pattern recognition techniques, such as hidden Markov modelling, are then applied to these features in order to generate the output transcriptions. Since the front end is designed to operate as a self-contained unit, it can be seen as a completely separate system. It can thus be treated as a "black box" whose output forms the input for the rest of the topic spotting system.

The training of the front end is done separately, using hand-transcribed data which does *not* have to be topic-specific.

1.1.3.2 Topic Score Generator

After the conversation has been transcribed, it is applied to the topic score generator. Various methods exist that can be applied to these transcriptions in order to generate a score for each of the topics. If need be, feature vectors are first extracted from the transcriptions and then used during the calculation of these scores.

The topic score generator is trained on topic-specific data. The topic spotter being implemented dictates whether hand- or machine-transcriptions are used. It is during this stage that useful keywords or keystrings are typically selected and their expected occurrence frequencies determined.

1.1.3.3 Recogniser

The recogniser can be used for either classification (i.e. a conversation can belong to only one topic) or detection (i.e. a conversation can belong to multiple topics). For classification problems, it will simply determine the most likely topic to be associated with the current conversation. However, for detection problems, the topic scores are compared to a threshold value in order to determine to which topics of interest the conversation should be allocated. If a conversation is correctly assigned to a topic, it is called a *detection*. However, if it is mistakenly pronounced as belonging to a topic, it is called a *false alarm*.

1.2 Research Focus

1.2.1 Problem Statement

The problem discussed in this thesis is one of detection rather than classification. A conversation can thus be allocated to one or more of a predefined set of topics which are defined by means of a number of non-overlapping transcribed example conversations, using phonemes as the most basic units.

Since it is a detection problem, system evaluation is performed by means of a receiver operating characteristic (ROC) curve, showing the trade-off between different false alarm- and detection-rates. The error area above the ROC curve is used to evaluate system performance.

Statistical significance of the results, represented by the ROC curves of any two topic spotting algorithms, is verified by means of the McNemar test [32, 33] modified to compensate for the fact that the task is one of detection.

1.2.2 Previous Work

In recent years various systems have emerged using phonemes rather than words as the most basic units. One of them is the Euclidean Nearest Wrong Neighbours (ENWN) algorithm [27, 28] which is based on an N-gram language modelling[3] [1] approach. Previous experiments [27, 28, 29] indicated that it outperforms other competing phoneme-based systems. However, in spite of ENWN's good performance, the algorithm is very basic. There is for example no stochastic modelling of the transcription errors introduced by the phoneme recognition front end, and it uses a simplistic distance measure when calculating the topic scores.

To address these issues, the new Stochastic Method for the Automatic Recognition of Topics (SMART) [24, 25, 26] was developed at the University of Stellenbosch, South Africa. It is an extension of ENWN, incorporating a statistical model of the recogniser performance and a probabilistically motivated distance measure for topic comparison. This results in robustness against phoneme recognition error and a corresponding improvement in performance.

Experiments [24, 25, 26] carried out on the Oregon Graduate Institute of Science and Technology's Multi-Language Telephone Speech (OGI-MLTS) Corpus [34] suggested that SMART yields a large improvement over the existing ENWN algorithm. It should, however, be emphasised that the database was rather small. This consequently implied that more rigorous testing had to be done on a much larger corpus.

1.2.3 Research Objectives

From the previous discussion it is clear that further research into the performance of SMART was needed. To this end, a number of objectives were defined. They are stated below:

[3] N represents the length of the keystrings for which the grammar is modelled.

- To implement a phoneme recogniser that would serve as a front end for the phoneme-based topic spotting systems being investigated.
- To optimise the implementation of both ENWN and SMART in the *PATREC* software system[4] in terms of computational efficiency, thereby making it practical to use on a very large corpus.
- To compare the two algorithms on the Switchboard Corpus[5] [35].
- To improve the accuracy of SMART even further.
- To write a paper in which the *new* comparative results between ENWN and SMART are outlined.

1.2.4 Contributions

The main contributions of this research are stated below:

- A practical, speaker independent, context independent phoneme recogniser was implemented having an overall recognition accuracy of 43.1%.
- The simulation times of ENWN and SMART were considerably reduced. ENWN's was reduced by 98.6%, while SMART's was reduced by 98.1%.
- ENWN and SMART were evaluated on the Switchboard Corpus. Subsequently, a substantial improvement of SMART over ENWN was observed, confirming the result previously obtained on the OGI-MLTS Corpus.
- An investigation was conducted into the possible improvement of SMART. This resulted in a new counting strategy, with a corresponding improvement in performance.

[4]This system has been developed by the Digital Signal Processing Group at the University of Stellenbosch, South Africa. It is a large collection of libraries written in C++ that can be used when doing signal processing, feature extraction, pattern recognition and statistical modelling.

[5]Switchboard is the *de facto* standard for evaluating topic spotting algorithms.

- A paper entitled "Phoneme-Based Topic Spotting on the Switchboard Corpus" [36], reporting on the comparative results between ENWN and SMART, was submitted and subsequently accepted for Eurospeech 2001.

1.3 Thesis Outline

ENWN will be introduced in Chapter 2. Possible reasons for ENWN's success are presented and the algorithm's weaknesses are pointed out. In addition to this, system operation is discussed in detail.

SMART is an extension of ENWN. Chapter 3 describes how the former extends the latter by introducing a model of phoneme recognition error and a probabilistically motivated distance measure.

Chapter 4 presents the approaches that were investigated in order to improve SMART's performance. Of particular interest is the new counting strategy of Section 4.4, since it is the only approach that resulted in SMART's amelioration.

The implementation of the phoneme recogniser is discussed in Chapter 5. It will be shown how signal processing-, feature extraction-, pattern recognition-, and statistical modelling-techniques were used to implement the front end.

Chapter 6 reports on the most important experiments that were conducted. The experimental setup, hardware specifications, and software implementation are also discussed.

In the final chapter, conclusions are drawn, and suggestions are given for further improving SMART's performance.

Chapter 2

Euclidean Nearest Wrong Neighbours Algorithm

Truth in science can be defined as the working hypothesis best suited to open the way to the next better one.

— Konrad Z. Lorenz

2.1 Introduction

The Centre de Recherche Informatique de Montréal (CRIM) recently proposed a computationally efficient phoneme-based topic spotting system. Closed-set[1] tests [27, 28, 29] indicated that their Euclidean Nearest Wrong Neighbours (ENWN) algorithm outperforms other competing phoneme-based methods. Not only does it outperform the other systems in terms of topic spotting performance, but in terms of speed as well. It also excelled in an open-set[2] scenario [27, 28]. However, in spite of ENWN's impressive performance, the algorithm is very basic. It has no probabilistic model to compensate for the errors

[1]The testing data contains topics that are present in the training data.

[2]The testing data contains topics that are *not* present in the training data. This situation is typically encountered in practice.

introduced by the phoneme recognition front end and it uses a primitive distance measure when calculating the topic scores. Why then does it work so well? Some of the possible reasons are listed below:

- The algorithm makes use of a trained lexicon containing hundreds or even thousands of short keystrings (phoneme n-grams). Although there is bound to be some redundancy, the sheer size of this lexicon compensates for the lack of sophistication of the individual keystrings.
- The keystrings in the trained lexicon are selected from a very large initial set giving the system the chance to choose those that are really useful.
- Discriminative training of the lexicon is done, meaning that emphasis is placed on the differences between topics.

This chapter discusses the internal workings of ENWN. An overview of the system is presented, followed by a detailed description of how it does feature extraction, topic comparison and detection. Finally, an explanation is given of how it is trained.

2.2 Description of the System

2.2.1 Overview

ENWN's system diagram is depicted in Figure 2.1 (adapted from Figure 3.1 in [24]). The core of the system is a large lexicon consisting of keystrings. The lexicon is initialised by selecting those phoneme n-grams that occur a fixed minimum number of times in the phonemic transcriptions of the training data and fall within a given length range.[3] The

[3]Phoneme n-grams can easily be extracted from a phoneme sequence; e.g. extracting 3-grams from the sequence "ABCDE" (A-E representing phonemes) yields "ABC", "BCD" and "CDE".

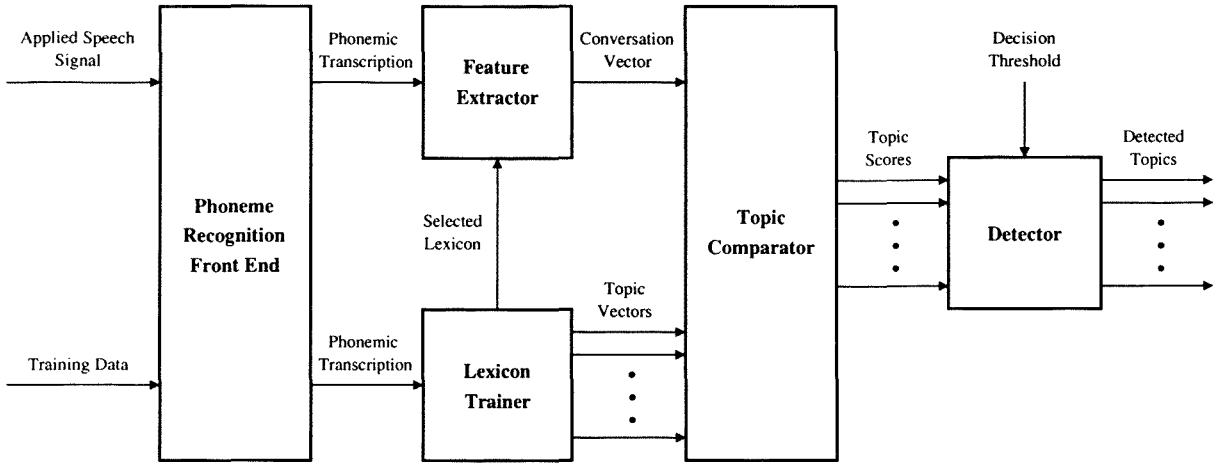


Figure 2.1: *System diagram of ENWN.*

lexicon is then pruned by iteratively removing lexicon members until an optimal size is reached. The criterion that is used to decide which members to remove is based on maximising the discrimination between topics. Once the final lexicon has been selected, it is used to extract the topic vectors, which characterise the topics of interest, from the training data.

When confronted with an applied test conversation, it is transcribed using the phoneme recognition front end. Subsequently, a conversation vector is constructed by measuring the occurrence frequency of each keystring in the lexicon. Afterwards, this vector is compared to the topic vectors using the Euclidean distance measure.[4] These distances are then normalised to sum to unity, producing the topic scores. Finally, these scores are thresholded in order to determine to which topics the conversation should be allocated.

2.2.2 Feature Extraction

Figure 2.2 (adapted from Figure 3.2 in [24]) illustrates the vector structures created by ENWN. To construct a conversation vector from an applied conversation's phonemic tran-

[4]Each topic's data is thus assumed to have an N-variate — N represents the total number of dimensions in the vector space — Gaussian distribution.

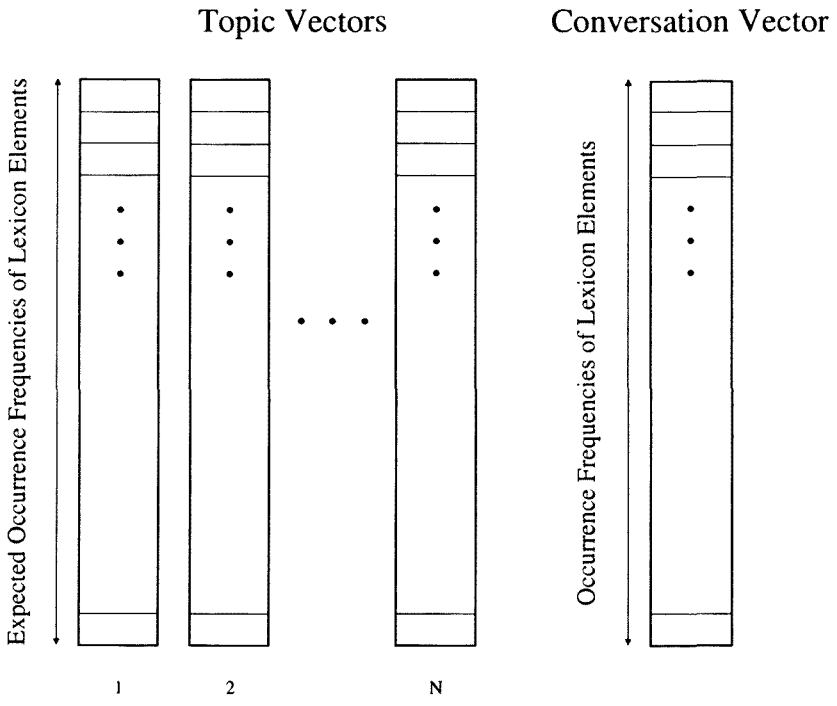


Figure 2.2: *Vector structures created by ENWN.*

scription, the occurrence frequency of each lexicon element has to be determined. It is accomplished by counting the number of times that the keystring occurs in the phoneme sequence and then dividing this integer value by the transcribed conversation's length (the total number of phonemes present in the phonemic transcription).

An example of how to generate the occurrence frequency of the keystring "g ae ey" is shown in Figure 2.3.[5] A window, equal in length to the size of the keystring, is placed on the left-hand side of the phoneme sequence. It is then moved forward, one phoneme at a time. While sliding the window, count the number of occurrences of the keystring. Afterwards, divide this number by the length of the transcribed conversation. A frequency of 0.13 ($\frac{2}{15}$) is thus obtained for the keystring of interest.

Each of the vectors shown on the left-hand side of Figure 2.2 is used to characterise a specific topic. If there are N topics, there will be N such feature vectors. A topic vector for a

[5]The syntax of the phonemes in this thesis is in accordance with the *ARPAbet* phonemic alphabet. For more information refer to Deller *et al.* [1], pp. 116–119.

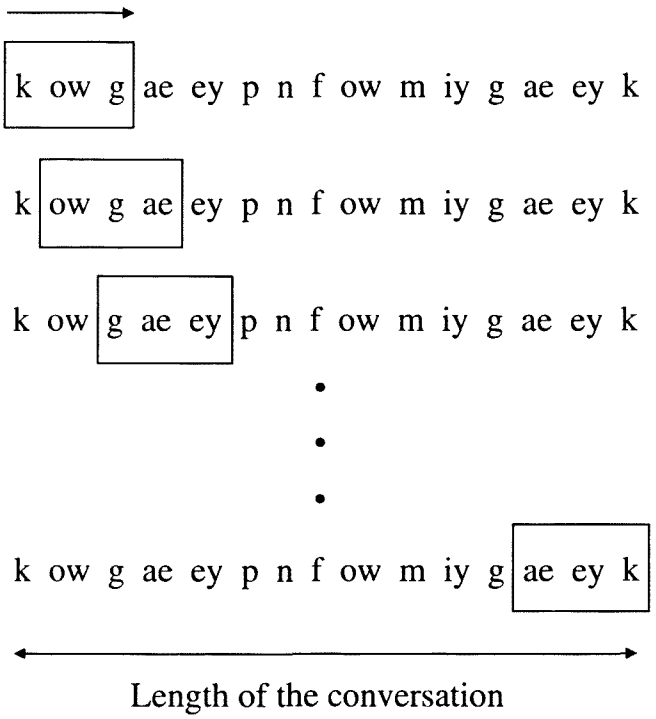


Figure 2.3: *Determining the occurrence frequency of a 3-gram in ENWN.*

given topic of interest can be obtained by simply averaging over all the conversation vectors belonging to that topic in the training data. As a result, a topic vector contains the *expected* occurrence frequencies of the lexicon members for the corresponding topic of interest.

2.2.3 Topic Comparison

After a conversation vector has been extracted from a test conversation's phonemic transcription, the *squared* Euclidean distance is calculated between it and each of the topic vectors. These distances are then normalised to sum to unity, producing the topic scores. These scores serve as an indication of how closely the conversation is related to each of the topics of interest.

2.2.4 Detection

The problem discussed here is one of detection. The topic scores are therefore thresholded in order to determine to which topics the conversation should be assigned. If a conversation is correctly allocated to a topic, it is called a *detection*. However, if it is mistakenly pronounced as belonging to a topic, it is called a *false alarm*. Take for example the situation presented in Figure 2.4. In this diagram:

- **C** represents the conversation vector,
- $\mathbf{T_1}$ to $\mathbf{T_{10}}$ represent the topic vectors, and
- n_1 to n_{10} represent the topic scores (i.e. the normalised squared Euclidean distances between the conversation vector and each of the topic vectors).

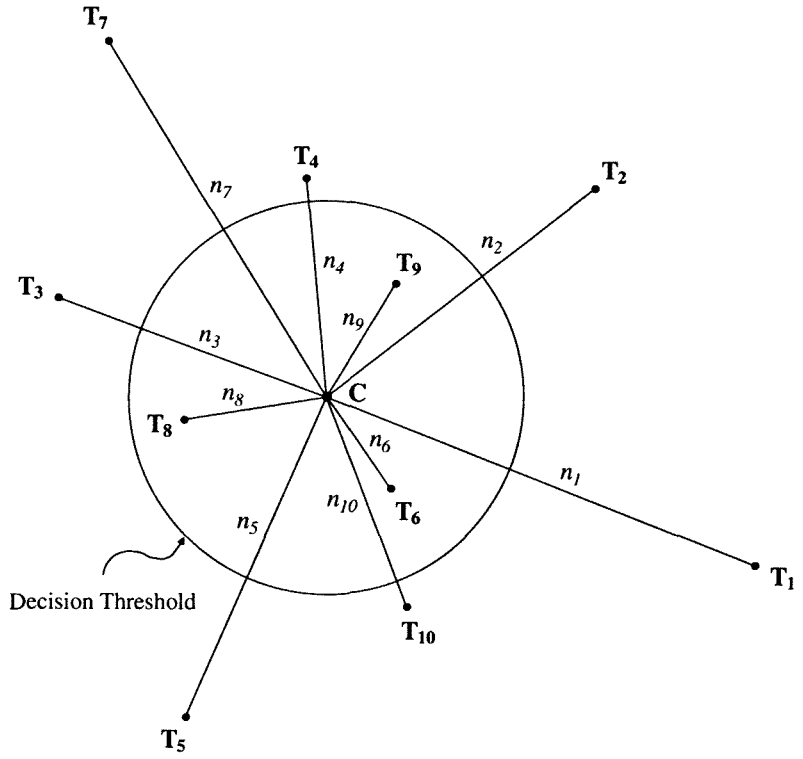


Figure 2.4: *A 2-dimensional representation of ENWN's detection process.*

If the conversation belongs to topic 8, there will be one detection (for topic 8) and two false alarms (for topics 6 and 9). On the other hand, if the conversation belongs to topic 2, there will be no detections and three false alarms (for topics 6, 8 and 9).

Since it is not desirable to evaluate system performance for a specific threshold value, it is rather evaluated by means of a receiver operating characteristic (ROC) curve, showing the trade-off between different false alarm- and detection-rates. During the evaluation process, the decision threshold is allowed to assume all of the values between 0 and 1. For each decision threshold the false alarm- and detection-rate can be calculated as follows:

- **Detection rate**: Divide the number of detections by the total number of possible detections.
- **False alarm rate**: Divide the number of false alarms by the total number of possible false alarms.

For the situation illustrated in Figure 2.4, with the conversation belonging to topics 1 and 6, the detection rate equals 0.5 ($\frac{1}{2}$) and the false alarm rate 0.25 ($\frac{2}{8}$).

2.2.5 Training

ENWN's lexicon is initialised by selecting those phoneme n-grams that occur a fixed minimum number of times in the phonemic transcriptions of the training data and fall within a given length range (a typical length range of 2 to 4 would imply using 2-, 3- and 4-grams). It is trained for use with the Euclidean distance measure by utilising the N-way[6] criterion [27, 28] to maximise discrimination between the topics. Note that when using this criterion, each training conversation must correspond to only one topic.

Let $\mathbf{C}$ represent the current training conversation vector, $\mathbf{R}(\mathbf{C})$ the correct (right) topic

[6] N represents the total number of topics that the system is trained to detect.

vector, $\mathbf{W}(\mathbf{C})$ the nearest wrong topic vector to $\mathbf{C}$, and $||\mathbf{C}, \mathbf{T}||_E$ the *squared* Euclidean distance between conversation vector $\mathbf{C}$ and topic vector $\mathbf{T}$. An error function is now defined as follows:

$$\begin{aligned} E(\mathbf{C}, \mathbf{R}(\mathbf{C}), \mathbf{W}(\mathbf{C})) &\stackrel{def}{=} ||\mathbf{C}, \mathbf{R}(\mathbf{C})||_E - ||\mathbf{C}, \mathbf{W}(\mathbf{C})||_E \\ &= \sum_i ((C_i - R_i)^2 - (C_i - W_i)^2), \end{aligned} \tag{2.1}$$

where the ith element of frequency vectors $\mathbf{C}$, $\mathbf{R}(\mathbf{C})$ and $\mathbf{W}(\mathbf{C})$ is indicated by C_i, R_i and W_i respectively. This error function is then accumulated over all training conversation vectors yielding a total error:

$$\begin{aligned} E_T &= \sum_{\mathbf{C}} E(\mathbf{C}, \mathbf{R}(\mathbf{C}), \mathbf{W}(\mathbf{C})) \\ &= \sum_{\mathbf{C}} \sum_i ((C_i - R_i)^2 - (C_i - W_i)^2) \\ &= \sum_i \sum_{\mathbf{C}} ((C_i - R_i)^2 - (C_i - W_i)^2). \end{aligned} \tag{2.2}$$

In order to maximise topic discrimination, E_T must be minimised. It thus follows that for each pruning iteration, the lexicon n-gram member making the largest positive contribution to E_T must be removed. Since the removal of a keystring has an impact on the feature vector space, each conversation vector's nearest wrong neighbour has to be redetermined before proceeding with the pruning of another lexicon element. The removal of keystrings will continue until a stopping criterion is met.

The extended training criterion [27, 28] is used to decide when to stop removing lexicon members. The training set is consequently split in the proportion 4:1 (training subset:validation subset). Once this has been accomplished, a lexicon is initialised from the training subset in the same way as the lexicon obtained from the full training set. A training run is then done on the training subset, evaluating system performance on the validation subset at different stages of the pruning process. Afterwards, the lexicon size giving optimal results is determined and a target percentage calculated by dividing this size

by the lexicon's initial size. The target percentage is subsequently used as a stopping criterion when training the lexicon obtained from the full training set. A flow diagram of the extended training algorithm is depicted in Figure 2.5 (adapted from Figure 3.3 in [24]), with the detailed flow diagram of the pruning step shown in Figure 2.6 (adapted from Figure 3.4 in [24]).

2.3 Summary

This chapter introduced the Euclidean Nearest Wrong Neighbours algorithm. The following conclusions can be drawn from the discussion:

- No probabilistic model exists to compensate for the errors introduced by the front end.
- Each topic's data is assumed to have an N-variate[7] Gaussian distribution with *unit* covariance matrix.

It should thus be apparent that an opportunity exists for improving system performance considerably. To accomplish this:

- The hard decisions made during the counting process can be replaced with a stochastic procedure that generates an expected count (also referred to as a *soft* count) for each of the lexicon elements.
- An advanced distance measure can be used to obtain a more realistic topic model.

Although experimental results [27, 28, 29] confirmed that ENWN outperforms competing phoneme-based topic spotters, it is obvious that further improvements should be possible.

[7] N represents the total number of dimensions in the vector space.

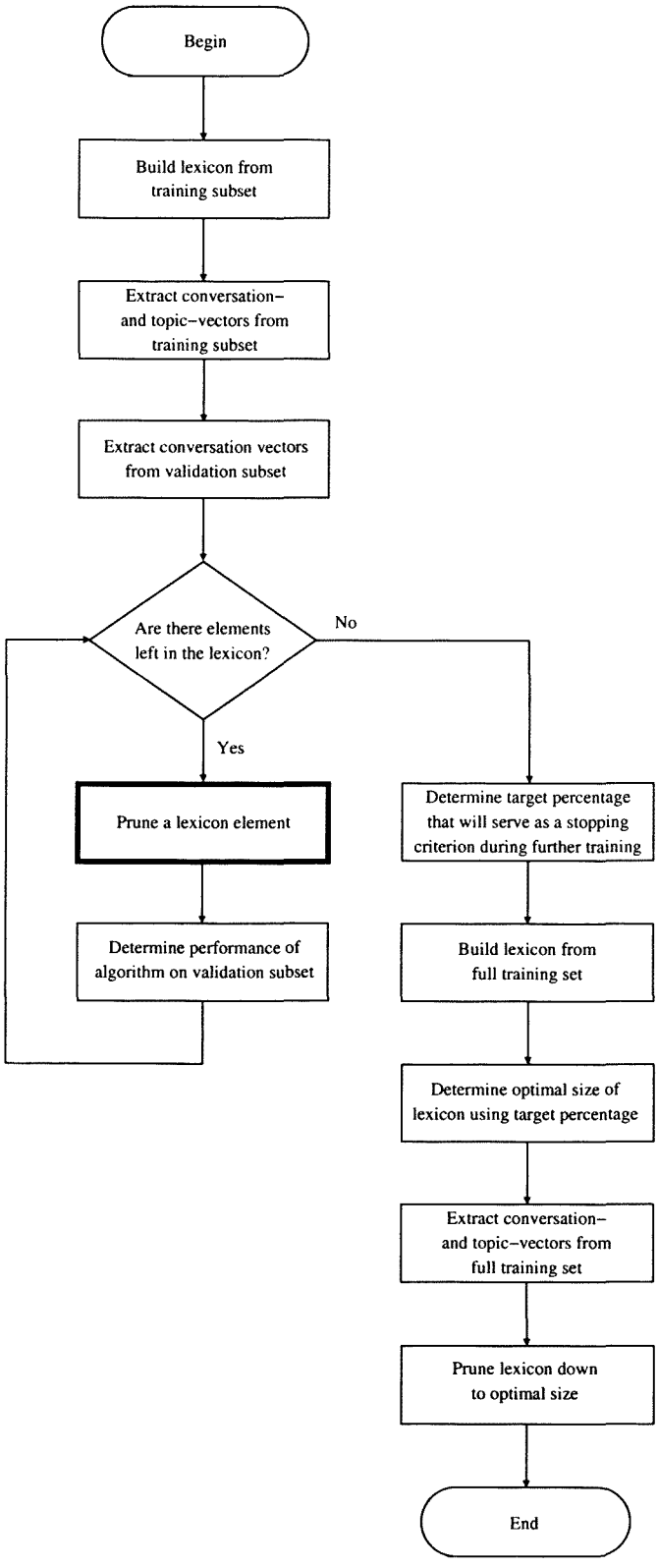


Figure 2.5: *Flow diagram of the extended training algorithm.*

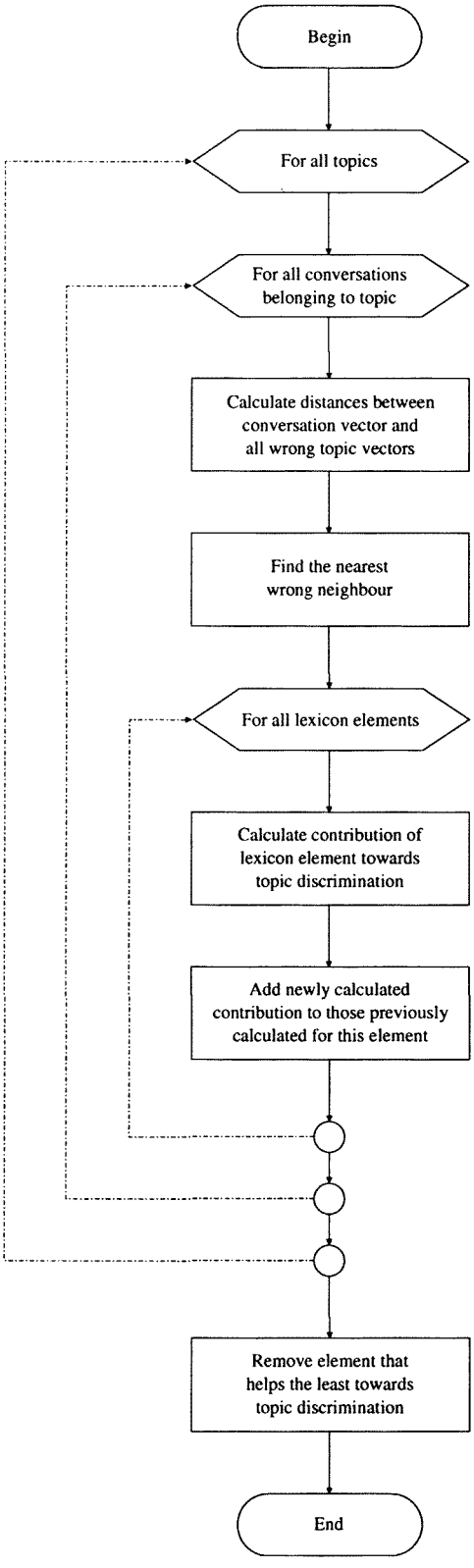


Figure 2.6: *Detail flow diagram of the pruning step.*

Chapter 3

Stochastic Method for the Automatic Recognition of Topics

There is no adequate defence, except stupidity, against the impact of a new idea.
— Percy W. Bridgeman

3.1 Introduction

When working with a phoneme-base topic spotting system, the front end can be seen as being responsible for the corruption of an applied conversation's true phoneme sequence. Sequence comparison theory dictates that three types of alterations occur during the transcription process [37], namely:

- **Insertions**: Phonemes are added to the sequence.
- **Deletions**: Phonemes are removed from the sequence.
- **Substitutions**: Certain members are replaced with new phonemes.

As a result, the same phonemic transcription can be generated for several different input conversations. To alleviate this problem, methods based on dynamic programming or hidden Markov models can be used. However, when working with large lexica, such as those produced by the Euclidean Nearest Wrong Neighbours algorithm, the simultaneous modelling of insertion-, deletion- and substitution-errors can become extremely computationally expensive. For practical applications, one would thus rather try to model only one of these errors.

ENWN is deterministic in the sense that it has no stochastic model to compensate for the errors introduced by the front end. It also uses a primitive distance measure when calculating the topic scores. To address these weaknesses, the Stochastic Method for the Automatic Recognition of Topics (SMART) [24, 25, 26] was developed at the University of Stellenbosch, South Africa. Although very similar in structure, it employs a more sophisticated keystring counting procedure which is based on a probabilistic model of the phoneme recogniser's substitution errors. Instead of simply counting the number of exact occurrences of each lexicon element (ENWN's approach), an *expected* count is obtained, taking into consideration that many of the keystrings in the phonemic transcriptions are corrupted. In addition to this, its topic comparator uses a probabilistically motivated distance measure. Consequently, these changes to ENWN result in robustness against phoneme recognition error and a corresponding improvement in performance.

In this chapter, details of the SMART algorithm will be presented. System operation is examined, followed by an explanation of how it is trained.

3.2 Description of the System

3.2.1 Overview

A system diagram of SMART is presented in Figure 3.1 (adapted from Figure 4.3 in [24]). From the figure it is obvious that its structure is nearly the same as ENWN's (Figure 2.1). The salient feature of this approach is a lexicon of *uncorrupted* keystrings which are assumed to occur in the true phoneme sequences of the training data. However, the lexicon is initialised in the same way as ENWN's. As a result, only those phoneme n-grams are included that occur a fixed minimum number of times in the training data's *corrupted* phonemic transcriptions and fall within a given length range. After initialisation, the lexicon is trained by iteratively removing those members that contribute the least towards topic discrimination. This process continues until a stopping criterion is met. Subsequent to the selection of the final lexicon, it is used to construct the topic vectors from the training data.

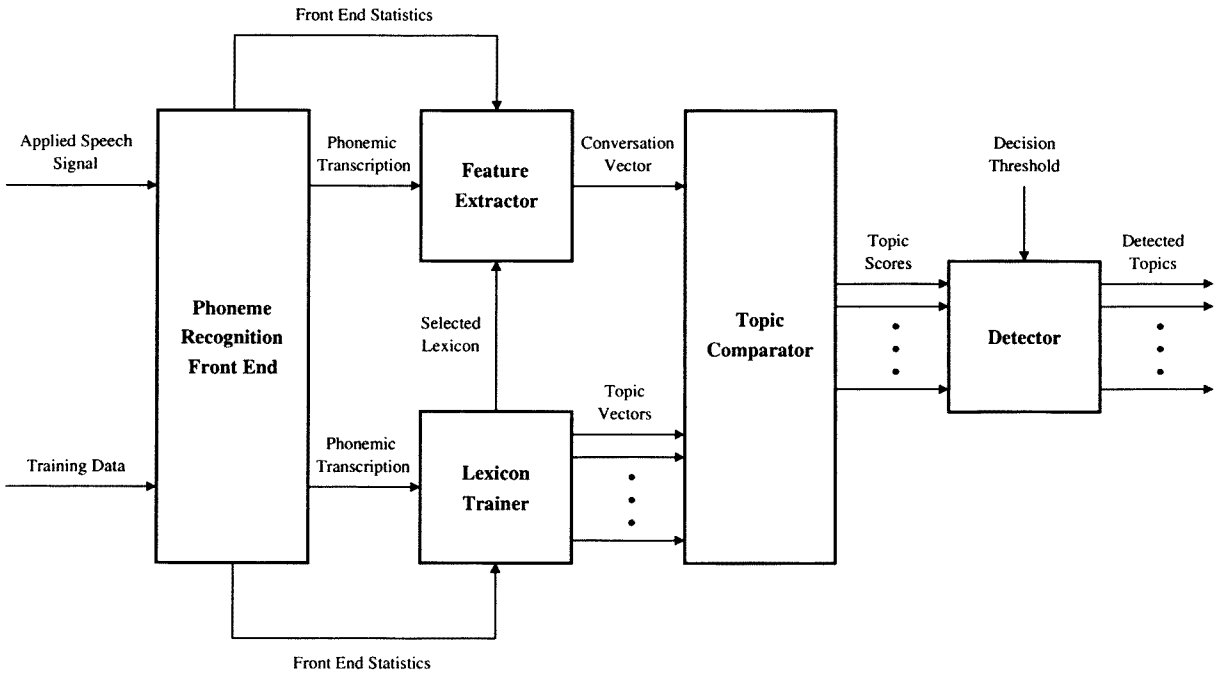


Figure 3.1: *System diagram of SMART.*

An applied test conversation is transcribed with the help of the phoneme recogniser. A conversation vector is then extracted by determining the occurrence frequency of each lexicon member in the *uncorrupted* phoneme sequence of the conversation. Afterwards, this vector is compared to each of the topic vectors using the cross-entropy distance measure. This will generate the topic scores that are compared to a threshold value in order to determine to which topics the conversation should be assigned.

3.2.2 Feature Extraction

The vector structures created by SMART are the same as those created by ENWN (Figure 2.2). However, to generate a conversation vector in SMART, a sophisticated counting procedure is employed to estimate the occurrence frequency of each lexicon element in the conversation's true phoneme sequence. A description of this process follows in Sections 3.2.2.1–3.2.2.4.

SMART's topic vectors are obtained in the same way as ENWN's. As a result, each topic vector is simply the statistical average of all training conversation vectors belonging to that topic.

3.2.2.1 Modelling the Front End

According to Scheffler *et al.* [25] a distinction must be made between an *uncorrupted* keystring $\mathbf{x}$ occurring in a conversation's true phoneme sequence and, corresponding to it, the *corrupted* keystring $\mathbf{y}$ observed at the output of the phoneme recogniser. The goal is to find the probability $P(\mathbf{x}|\mathbf{y})$ of an *uncorrupted* keystring given the *corrupted* one. Since only substitution errors are modelled by SMART, a one-to-one correspondence between the phonemes in these keystrings is assumed. Under this assumption, the structure illustrated in Figure 3.2 (Figure 4.2 in [24]) can be used to perform the estimate.

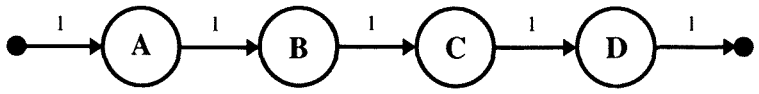


Figure 3.2: *Structure of the sequence matcher.*

A sequence matcher[1] allows only one state sequence, with all transition probabilities equal to 1. The states **A** to **D** correspond to the phonemes of **x** (illustrated for a keystring of length 4). An observed keystring **y** can thus be matched to **x** by simply matching each phoneme in **y** to the corresponding state in the sequence matcher. Note that **y** is constrained to have the same length as **x** (an effect of neglecting insertions and deletions).

Given phoneme y_i, observed at position i in the *corrupted* keystring, the ith state of the model approximates the probability $P(x_i|y_i)$ that y_i was produced as a result of phoneme x_i in the original keystring. The desired probability is estimated from a context independent confusion matrix describing the front end's performance (Section 5.6):

$$\begin{aligned} P(x_i|y_i) &= \frac{P(x_i, y_i)}{P(y_i)} \\ &\simeq \frac{Conf(y_i, x_i)}{\sum_j Conf(y_i, q_j)}, \end{aligned} \tag{3.1}$$

where *Conf(a,b)* (a and b are phonemes) is the entry in the confusion matrix corresponding to row a and column b.[2] q_j represents the jth phoneme in the phonemic alphabet.

Ignoring context dependencies, the sequence matcher produces an output by combining the scores of the individual states:

$$\begin{aligned} P(\mathbf{x}|\mathbf{y}) &= \prod_i P(x_i|y_i) \\ &\simeq \prod_i \frac{Conf(y_i, x_i)}{\sum_j Conf(y_i, q_j)}. \end{aligned} \tag{3.2}$$

[1] A sequence matcher is similar to a discrete HMM. However, whereas a sequence matcher produces the probability $P(\mathbf{x}|\mathbf{y})$, a discrete HMM produces the probability $P(\mathbf{y}|\mathbf{x})$.

[2] The rows correspond to the classified phonemes, while the columns correspond to the original input phonemes.

3.2.2.2 Soft Counts of Lexicon Members

Using the method described above, the probability of each lexicon member $\mathbf{x}$ matching a section of the conversation's *uncorrupted* phoneme sequence is estimated. The expected value of the "true" count of $\mathbf{x}$, $E(Cnt(\mathbf{x}))$, can therefore be calculated by summing these probabilities over all subsets of the *corrupted* phonemic transcription:

$$E(Cnt(\mathbf{x})) = \sum_{n} P(\mathbf{x}|\mathbf{y}_n), \tag{3.3}$$

where $\mathbf{y}_n$ is the nth keystring in the conversation's observed phoneme sequence. This expected value is a *soft* count which yields an estimate of the number of occurrences of a keystring and eliminates the hard decisions made by ENWN during the counting process.

3.2.2.3 Frequencies of Lexicon Members

After obtaining the soft count of a lexicon member, it must be divided by the length of the *corrupted* phonemic transcription in order to generate the occurrence frequency of the keystring in the *uncorrupted* phoneme sequence. This length is used, since no other realistic estimate exists of the conversation's true phoneme sequence.

3.2.2.4 Example

An example of how to generate the occurrence frequency of the *uncorrupted* keystring $\mathbf{x}$, represented by "f iy ey", is shown in Figure 3.3. A window, equal in length to the size of the keystring, is placed on the left-hand side of the transcribed conversation and is then moved forward, one phoneme at a time. While sliding the window, use a sequence matcher (Equation 3.2) to calculate the probability of the *corrupted* keystring $\mathbf{y}$ being the *uncorrupted* one, and then add this result to the previously calculated probabilities (Equation 3.3). Afterwards, the keystring's occurrence frequency is generated by dividing

this soft count by the length of the transcribed conversation which is 5.

The first step is therefore to determine the keystring's soft count using Equations 3.2 and 3.3 (the performance of the phoneme recognition front end is described by the context independent confusion matrix[3] of Figure 3.4):

$$\begin{aligned} E(Cnt(\mathbf{x})) &= \sum_{n=1}^{3} P(\mathbf{x}|\mathbf{y}_n) \\ &\simeq \sum_{n=1}^{3}\prod_{i=1}^{3} \frac{Conf(y_{in}, x_i)}{\sum_j Conf(y_{in}, q_j)} \\ &= \frac{8}{41}\frac{5}{43}\frac{2}{33} + \frac{1}{43}\frac{1}{33}\frac{14}{43} + \frac{5}{33}\frac{3}{43}\frac{3}{61} \\ &= 2.12 \times 10^{-3}. \end{aligned} \tag{3.4}$$

This number is now divided by the length of the transcribed conversation:

$$\begin{aligned} F(\mathbf{x}) &= \frac{2.12 \times 10^{-3}}{5} \\ &= 424 \times 10^{-6}, \end{aligned} \tag{3.5}$$

where $F(\mathbf{x})$ represents the occurrence frequency of "f iy ey". A nonzero frequency of occurrence is thus obtained. This is in contrast to ENWN's hard counting strategy which would have produced an occurrence frequency of 0.

3.2.3 Topic Comparison

After a conversation vector has been extracted from the phonemic transcription of an applied test conversation, it is compared to the topic vectors as follows:

- The cross-entropy (refer to Section 3.2.3.1 for a derivation of this distance measure) is calculated between the conversation vector and each of the topic vectors.

[3]The phonemic alphabet is assumed to contain only those phonemes shown in the confusion matrix.

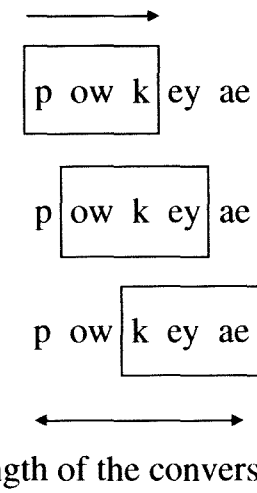


Figure 3.3: *Determining the occurrence frequency of a 3-gram in SMART.*

Classified phoneme \ Input phoneme	ey	f	iy	k	ow	p	ae	Total for each row
ae	3	3	7	9	4	5	***30***	61
p	7	***8***	3	5	2	***14***	2	41
ow	5	1	***5***	2	***18***	7	5	43
k	2	5	1	***11***	1	7	6	33
iy	4	3	***25***	6	3	2	7	50
f	4	***22***	5	4	3	4	8	50
ey	***14***	8	3	6	7	1	4	43

Figure 3.4: *Front end's context independent confusion matrix.*

- These distances are then normalised to sum to unity.

This produces the topic scores which serve as an indication of how closely the conversation is related to each of the topics of interest.

3.2.3.1 Cross-Entropy Distance Measure

It is natural to think of the frequency vector of each topic as describing a partial topic-dependent unigram language model. This is because each frequency is in fact a maximum likelihood estimate of the context independent probability of occurrence for the corresponding keystring. Adopting this point of view, the probability $P(\mathcal{C}|\mathcal{T})$ of conversation $\mathcal{C}$ given topic $\mathcal{T}$ can be calculated as follows:

$$P(\mathcal{C}|\mathcal{T}) = \prod_i P(L_i|\mathcal{T})^{Cnt(L_i)}, \tag{3.6}$$

where $P(L_i|\mathcal{T})$ is the probability of observing the ith lexicon element L_i given topic $\mathcal{T}$. $Cnt(L_i)$ is the number of times that this lexicon element occurs in the true phoneme sequence of the conversation. These quantities are approximated using the ith elements of conversation vector $\mathbf{C}$ and topic vector $\mathbf{T}$ (C_i and T_i respectively):

$$P(\mathcal{C}|\mathcal{T}) \simeq \prod_i T_i^{NC_i}, \tag{3.7}$$

where N is the number of phonemes in the conversation's *corrupted* phonemic transcription. By taking the negative logarithm, Equation 3.7 becomes:

$$-\log P(\mathcal{C}|\mathcal{T}) = -\sum_i NC_i \log T_i. \tag{3.8}$$

After Bayesian inversion, Equation 3.8 can be written as:

$$-\log P(\mathcal{T}|\mathcal{C}) = -\sum_i NC_i \log T_i - \log(\frac{P(\mathcal{T})}{P(\mathcal{C})}). \tag{3.9}$$

Since the occurrence frequency of a topic in the training data cannot be assumed to correlate with that in the testing data of a practical application, it is standard practice to assume equal prior topic probabilities. The second term on the right-hand side of Equation 3.9 is consequently discarded. In addition to this, the constant scaling factor N is ignored. This may be done, since the distances are normalised when producing the topic scores. As a result, the cross-entropy (CE) [38] between vectors $\mathbf{C}$ and $\mathbf{T}$ remains. The CE distance measure is therefore defined as follows:

$$||\mathbf{C}, \mathbf{T}||_{CE} \stackrel{def}{=} -\sum_i C_i \log T_i. \tag{3.10}$$

3.2.4 Detection

The topic scores are thresholded in order to determine to which topics the conversation should be allocated. Evaluation is done by means of an ROC curve with the decision threshold varying between 0 and 1. For an in-depth discussion regarding the detection process refer to Section 2.2.4, keeping in mind that SMART uses the cross-entropy distance measure rather than the Euclidean norm.

3.2.5 Training

SMART's lexicon is initialised in the same way as ENWN's. As a result, only those phoneme n-grams are included that occur a fixed minimum number of times in the training data's transcribed conversations and fall within a given length range. However, these keystrings are assumed to correspond to phoneme n-grams found in the true phoneme sequences of the training data. Although initialisation of the lexicon from keystrings that occur in the transcribed training conversations is not ideal, it is hoped that the distribution of the keystrings in the *corrupted* data will not differ much from those found in the *uncorrupted* data.

After initialisation of the lexicon, it is trained for use with the new CE distance measure by utilising the N-way[4] criterion to maximise topic discrimination. Consequently, using the notation introduced in Section 2.2.5, the error function of Equation 2.1 becomes:

$$\begin{aligned} E(\mathbf{C},\mathbf{R}(\mathbf{C}),\mathbf{W}(\mathbf{C})) &\stackrel{def}{=} ||\mathbf{C},\mathbf{R}(\mathbf{C})||_{CE} - ||\mathbf{C},\mathbf{W}(\mathbf{C})||_{CE} \\ &= -\sum_i C_i \log R_i + \sum_i C_i \log W_i \\ &= \sum_i (C_i \log W_i - C_i \log R_i). \end{aligned} \tag{3.11}$$

Accumulating this error function over all training conversation vectors yields the following total error:

$$\begin{aligned} E_T &= \sum_{\mathbf{C}} E(\mathbf{C},\mathbf{R}(\mathbf{C}),\mathbf{W}(\mathbf{C})) \\ &= \sum_{\mathbf{C}} \sum_i (C_i \log W_i - C_i \log R_i) \\ &= \sum_i \sum_{\mathbf{C}} (C_i \log W_i - C_i \log R_i). \end{aligned} \tag{3.12}$$

E_T must now be minimised in order to maximise topic discrimination. This is accomplished by using the extended training algorithm (Section 2.2.5).

3.3 Summary

This chapter presented SMART as an extension of ENWN. It was shown how SMART combats the effects of sequence corruption through the use of a sophisticated keystring counting procedure that is based on a probabilistic model of the front end's substitution errors. In addition to this, the introduction of the CE distance measure was discussed.

Upon initial implementation [24, 25, 26], SMART was found to perform substantially better than ENWN on the topic-specific section of the OGI-MLTS Corpus. Closed-set tests

[4] N represents the total number of topics that the system is trained to detect.

revealed that the improvement of SMART over ENWN is characterised by a 26% reduction in ROC error area. However, the limited amount of data available and the short conversation lengths (approximately 10 seconds each) suggested that more rigorous testing was required.

In this research, the algorithms were therefore re-implemented to run on the much larger Switchboard Corpus. Subsequently, a substantial improvement of SMART over ENWN was observed (Section 6.5.1), confirming the result that was previously obtained. These modifications to ENWN consequently result in SMART being a superior topic spotting system.

Chapter 4

Improving SMART

We are continually faced with a series of great opportunities brilliantly disguised as insoluble problems.

— John W. Gardner

4.1 Introduction

SMART performs substantially better than ENWN. Nevertheless, numerous techniques remain that can possibly improve SMART's performance. In his seminal work [24], Scheffler proposed several approaches, some of which were evaluated during his research. However, except for one instance, no improvement was observed. Since a rather small corpus was used during the evaluation process, it was decided to repeat *all* the experiments on the Switchboard Corpus in order to confirm Scheffler's findings.

The techniques employed by Scheffler to try and improve the performance of SMART will be covered in the first part of this chapter. Afterwards, a description is given of a new soft counting strategy that was developed during this research. Note that closed-set experiments were conducted to evaluate the different approaches.

4.2 Topic Models

4.2.1 Parametric Distribution Modelling

The parametric topic models that were evaluated by Scheffler are listed below (in each case the dimensions are assumed to be statistically independent):

- **N-variate Gaussian model with diagonal covariance matrix**: When using the Euclidean distance measure it is assumed that the data's distribution for each topic:
  - is symmetric,
  - decreases monotonically from a central maximum,
  - has equal variance on each of the dimensions, and
  - that the covariances between the dimensions are zero.

  For this model to be successful, the data must therefore have an N-variate[1] Gaussian distribution with *unit* covariance matrix. However, this situation rarely presents itself in practice. A better approach would thus be to at least estimate the variance on each of the dimensions. As a result, the N-variate Gaussian distribution model with *diagonal* covariance matrix can be used to obtain a better estimate of the data's distribution.

- **Beta model**: Each topic's data points are located within a unit hyper-sphere which is centred at the origin of the Cartesian plane. The beta model can thus be used for modelling purposes, since its input values lie within the same hyper-sphere.

- **Exponential model**: The data points for each topic are primarily located close to the origin of the axes. Consequently, an exponential model of decay can be employed with its maximum value located at the origin. The main advantage of using this

[1] N represents the total number of dimensions in the vector space.

approach is that no variance estimates are necessary. The possible deterioration in topic spotting performance that could result from the incorrect estimation of the variance on each of the dimensions is therefore avoided.

Replacement of the CE distance measure with these *unimodal* parametric distribution models yielded a deterioration in performance.[2] This was also observed by Scheffler. The most likely reason for this is that the data's distribution is *multimodal.*

4.2.2 Minkowski Metric

There are various notions of length. For example, the Minkowski metric of order s (also called the l_s metric) [1] which is defined as follows:

$$||\mathbf{v}||_s = \left(\sum_i |v_i|^s\right)^{\frac{1}{s}} \quad with\ s \in Z, \tag{4.1}$$

where $\mathbf{v}$ represents a multi-dimensional feature vector. v_i represents the ith element of vector $\mathbf{v}$.

In practice only three of these metrics are usually employed, namely:

- The l_1 or *city block* metric:

$$||\mathbf{v}||_1 = \sum_i |v_i|. \tag{4.2}$$

[2]When replacing the CE distance measure, SMART's training algorithm must be adapted for the new topic scoring criterion in order to maximise system performance.

- The l_2 or *Euclidean* metric:

$$||\mathbf{v}||_2 = \left(\sum_i |v_i|^2\right)^{\frac{1}{2}}. \tag{4.3}$$

- The l_∞ or *Chebyshev* metric (corresponding to the Minkowski metric as $s \to \infty$):

$$||\mathbf{v}||_\infty = \max_i |v_i|. \tag{4.4}$$

Previous experiments confirmed that the CE distance measure outperforms the squared l_2 metric (ENWN's distance measure). As a result, some cases for $s \neq 2$ had to be checked. Ignoring the Chebyshev metric, the l_1 and cubed l_3 metrics were thus evaluated.[3] However, no improvement in topic spotting performance was measured. This confirmed the results obtained by Scheffler.

4.3 Excluding Garbage Classes from Keystrings

Scheffler postulated that excluding the garbage class representing non-speech sounds and silence (Section 5.3.2) from the lexicon members might improve performance, reasoning that the attachment of semantic meaning to this garbage class is counter-intuitive. He subsequently conducted experiments where each occurrence of it was treated as a break in the conversation. Results indicated that his hypothesis might be correct, since an improvement was observed. However, re-evaluation of this approach on the Switchboard Corpus resulted in a deterioration in performance.

In this research, another garbage class was defined representing stop consonant closures and releases occurring without the other (Section 5.3.2). Inspection of the hand-transcribed phonemic transcriptions employed for the training and testing of the phoneme recognition

[3]When using a Minkowski metric as a distance measure, the N-way criterion dictates that its root should not be taken.

front end, revealed that it regularly occurs on word boundaries.[4] It was consequently treated as a break in the conversation in order to prevent the formation of keystrings across these boundaries. Experiments were subsequently conducted to determine whether its exclusion from the lexicon members would result in SMART's amelioration. However, no improvement was observed.

The exclusion of both of these garbage classes from the lexicon members was also investigated, with no improvement in topic spotting performance.

4.4 Refinement of SMART's Soft Counting Strategy

In Section 3.2.2, SMART's soft counting strategy was introduced. It was shown how this counting procedure relies on Equations 3.2 and 3.3 to determine the "true" count of each lexicon element in the conversation's *uncorrupted* phoneme sequence. Although excellent results were obtained with it, much could still be gained if a closer connection could be established with the front end. SMART's soft counting strategy was therefore refined during this research to include low level information supplied by the phoneme recogniser.

4.4.1 Modelling the Front End

Let $\mathbf{x}$ represent an *uncorrupted* keystring occurring in the conversation's true phoneme sequence, $\mathbf{y}$ a *corrupted* keystring (having the same length as $\mathbf{x}$) observed at the output of the phoneme recogniser, and $\mathbf{z}$ the multi-dimensional stream of acoustic feature vectors (Section 5.2.2) on which $\mathbf{y}$ is based. The goal is to find the probability $P(\mathbf{x}|\mathbf{z})$ of an *uncorrupted* keystring given the acoustic feature vectors of the *corrupted* one. Since no insertions and deletions are to be modelled, the sequence matcher in Figure 3.2 can be used to perform the estimate. However, instead of matching $\mathbf{y}$ to the sequence matcher,

[4] 64.5% of the total number of occurrences of this garbage class occur on word-boundaries.

rather match the acoustic feature vectors of each phoneme in $\mathbf{y}$ to the corresponding state of the sequence matcher.

Given the acoustic feature vectors $\mathfrak{z}_i$, corresponding to phoneme y_i which is observed at position i in the *corrupted* keystring, the ith state of the model approximates the probability $P(x_i|\mathfrak{z}_i)$ that $\mathfrak{z}_i$ was produced as a result of the phoneme x_i in the original keystring. Assuming that the kth phoneme in the phonemic alphabet is represented by q_k (modelled as a standard three-state left-to-right one-skip HMM — see Section 5.4), the probability can be calculated as follows:

$$\begin{aligned} P(x_i|\mathfrak{z}_i) &= \sum_k P(x_i, q_k|\mathfrak{z}_i) \\ &= \sum_k P(x_i|q_k, \mathfrak{z}_i)P(q_k|\mathfrak{z}_i) \\ &\simeq \sum_k P(x_i|q_k)P(q_k|\mathfrak{z}_i). \end{aligned} \tag{4.5}$$

$P(x_i|q_k)$ is estimated using the context independent confusion matrix describing the front end's performance (Section 5.6):

$$\begin{aligned} P(x_i|q_k) &= \frac{P(x_i, q_k)}{P(q_k)} \\ &\simeq \frac{Conf(q_k, x_i)}{\sum_j Conf(q_k, q_j)}, \end{aligned} \tag{4.6}$$

where *Conf(a,b)* (a and b are phonemes) is the entry in the confusion matrix corresponding to row a and column b.[5]

$P(q_k|\mathfrak{z}_i)$ (Equation 4.5) represents the *a posteriori* probability [39] of observing q_k given y_i's acoustic observation sequence $\mathfrak{z}_i$. Its value can be determined using Bayes' theorem:

$$P(q_k|\mathfrak{z}_i) = \frac{f(\mathfrak{z}_i|q_k)P(q_k)}{f(\mathfrak{z}_i)}, \tag{4.7}$$

[5]The rows correspond to the classified phonemes, while the columns correspond to the original input phonemes.

where $f(\mathfrak{z}_i|q_k)$ is the value of the probability density function (pdf) which is calculated by the phoneme-HMM of q_k for $\mathfrak{z}_i$. $P(q_k)$ is the *a priori* probability [39] of phoneme k, and is estimated by simply determining its frequency of occurrence in the front end's training data. The normalising pdf $f(\mathfrak{z}_i)$ is the same for all the phoneme classes and therefore causes an arbitrary, but *common* scaling of the *a posteriori* probabilities. Using the marginal density theorem [39]:

$$\begin{aligned} P(q_k|\mathfrak{z}_i) &= \frac{f(\mathfrak{z}_i|q_k)P(q_k)}{\sum_j f(\mathfrak{z}_i, q_j)} \\ &= \frac{f(\mathfrak{z}_i|q_k)P(q_k)}{\sum_j f(\mathfrak{z}_i|q_j)P(q_j)}. \end{aligned} \tag{4.8}$$

Equations 4.6 and 4.8 are now substituted into Equation 4.5:

$$P(x_i|\mathfrak{z}_i) \simeq \sum_k \frac{Conf(q_k, x_i)}{\sum_j Conf(q_k, q_j)} \frac{f(\mathfrak{z}_i|q_k)P(q_k)}{\sum_j f(\mathfrak{z}_i|q_j)P(q_j)}. \tag{4.9}$$

This formula is subsequently used to obtain a score for each sequence matcher state.

Ignoring context dependency, the desired probability P($\mathbf{x}|\mathbf{z}$) of a specific keystring $\mathbf{x}$ given the acoustic feature vectors $\mathbf{z}$ is determined by combining the scores of the individual states:

$$\begin{aligned} P(\mathbf{x}|\mathbf{z}) &= \prod_i P(x_i|\mathfrak{z}_i) \\ &\simeq \prod_i \sum_k \frac{Conf(q_k, x_i)}{\sum_j Conf(q_k, q_j)} \frac{f(\mathfrak{z}_i|q_k)P(q_k)}{\sum_j f(\mathfrak{z}_i|q_j)P(q_j)}. \end{aligned} \tag{4.10}$$

4.4.2 Soft Counts of Lexicon Members

The expected "true" count of lexicon member $\mathbf{x}$ in the conversation's *uncorrupted* phoneme sequence, $E(Cnt(\mathbf{x}))$, can be calculated by summing these probabilities (Equation 4.10)

over all subsets of the transcribed conversation:

$$E(Cnt(\mathbf{x})) = \sum_{n} P(\mathbf{x}|\mathbf{z}_n), \tag{4.11}$$

where $\mathbf{z}_n$ is the acoustic feature vectors corresponding to $\mathbf{y}_n$ which is the nth keystring in the observed phonemic transcription.

4.4.3 Frequencies of Lexicon Members

The occurrence frequency of lexicon member $\mathbf{x}$ is obtained in the same way as described in Section 3.2.2.3.

4.4.4 Example

Reconsider the scenario presented in Section 3.2.2.4. This time, however, calculate the occurrence frequency of the *uncorrupted* keystring "f iy ey" ($\mathbf{x}$) using the newly developed counting strategy. The first step is consequently to determine its soft count. A window, having the same length as the keystring, is therefore placed on the left-hand side of the conversation's *corrupted* phonemic transcription and is then moved forward one phoneme at a time (Figure 3.3). While sliding the window, use the modified sequence matcher (Equation 4.10) to calculate the probability of the *uncorrupted* keystring, given the acoustic feature vectors $\mathbf{z}$ of the *corrupted* keystring $\mathbf{y}$. These probabilities are then accumulated over all subsets of the *corrupted* phoneme sequence in order to obtain the soft count of $\mathbf{x}$ (Equation 4.11). Afterwards, divide this count by the length of the *corrupted* phonemic transcription in order to generate the keystring's frequency of occurrence.

The following soft count is thus obtained (the *a posteriori* probabilities are given in Figure 4.1, while the context independent confusion matrix describing the front end's per-

formance is reproduced in Figure 4.2 for convenience):

$$\begin{aligned} E(Cnt(\mathbf{x})) &= \sum_{n=1}^{3} P(\mathbf{x}|\mathbf{z}_n) \\ &\simeq \sum_{n=1}^{3}\prod_{i=1}^{3}\Big(\sum_{k=1}^{7}\frac{Conf(q_k,x_i)}{\sum_{j=1}^{7}Conf(q_k,q_j)}P(q_k|\mathfrak{z}_{in})\Big) \\ &= (0.196)(0.134)(0.067)+(0.115)(0.032)(0.194)+ \\ &\quad (0.153)(0.154)(0.055) \\ &= 3.77\times 10^{-3}. \end{aligned} \tag{4.12}$$

Dividing this number by the length of the transcribed conversation yields the keystring's occurrence frequency, $F(\mathbf{x})$:

$$\begin{aligned} F(\mathbf{x}) &= \frac{3.77\times 10^{-3}}{5} \\ &= 754\times 10^{-6}, \end{aligned} \tag{4.13}$$

as compared to 424×10^{-6} in Section 3.2.2.4.

4.4.5 Discussion

Upon implementation of the new soft counting strategy, a slight improvement in topic spotting performance was observed (a formal presentation of this result can be found in Section 6.5.2). However, closer inspection of the front end's phoneme-HMMs revealed that they behave "pigheadedly". As a result, the *a posteriori* probability of the winning phoneme will usually be close to one.[6] When this happens, Equations 3.2 and 4.10 are equivalent. The slight improvement in performance can therefore be attributed to the small number of instances when the phoneme-HMMs did not behave in this manner. Much could thus

[6]The *a posteriori* probability of the winning phoneme in CRIM's subset of the Switchboard Corpus that was used for evaluation purposes (Section 6.2.1), was above 0.98 in 70.1% of the cases.

Conversation

A posteriori probabilities

	p	ow	k	ey	ae
ae	0.04	0.21	0	0.21	0.89
p	0.67	0.05	0.03	0.16	0
ow	0	0.3	0	0.02	0.06
k	0.17	0	0.96	0	0
iy	0.04	0.09	0	0.17	0.05
f	0.08	0.07	0	0.01	0
ey	0	0.28	0.01	0.43	0

Figure 4.1: *A posteriori probabilities for the transcribed conversation.*

Total for each row

Classified phoneme

ae			7	9	4	5	***30***
p			3	5	2	***14***	2
ow			5	2	***18***	7	5
k			1	***11***	1	7	6
iy			***25***	6	3	2	7
f			5	4	3	4	8
ey			3	6	7	1	4
	ey	**f**	**iy**	**k**	**ow**	**p**	**ae**

61
41
43
33
50
50
43

Input phoneme

Figure 4.2: *Front end's context independent confusion matrix.*

be gained if this situation is remedied.

4.5 Summary

Various approaches were evaluated during this research in order to try and improve SMART's performance. They are listed below:

- Using a new topic model:
  - N-variate Gaussian model with diagonal covariance matrix,
  - beta model,
  - exponential model, and
  - Minkowski metrics.
- Excluding garbage classes from lexicon members.
- Refining its soft counting strategy.

Of these, only the new soft counting strategy of Section 4.4 resulted in SMART's amelioration. However, many opportunities still exist for improving SMART. To this end, a number of approaches are suggested in Section 7.2. By concentrating on these points, it is hoped that better topic spotting performance might be obtained.

Chapter 5

Implementation of the Phoneme Recognition Front End

. . . the structure of language has aroused man's curiosity for all recorded time. The reason?—because it is not obvious. Meeting a stranger, one can make a fair guess at the meanings of his gestures and facial expressions. . . . But the codification of language creates a sort of mystery where guessing is of little use.

— Bolinger, 1981; 17

5.1 Introduction

In this research, telephone conversations had to be phonemically transcribed in order to allow the inter-comparison of the topic spotting algorithms. A phoneme recogniser was therefore implemented to serve as a front end for these topic spotters. It was designed to operate as a self-contained unit and can thus be seen as a completely separate system whose output forms the input for the rest of the topic spotting system. Unless otherwise noted, the implementation of the front end was accomplished with the help of the *PATREC* software system.

The implementation of the phoneme recogniser is discussed in this chapter. An overview of the phonemic transcription process is presented, followed by an explanation of how it was trained and evaluated.

5.2 Overview of the Phonemic Transcription Process

A block diagram of the implemented phoneme recogniser can be seen in Figure 5.1.

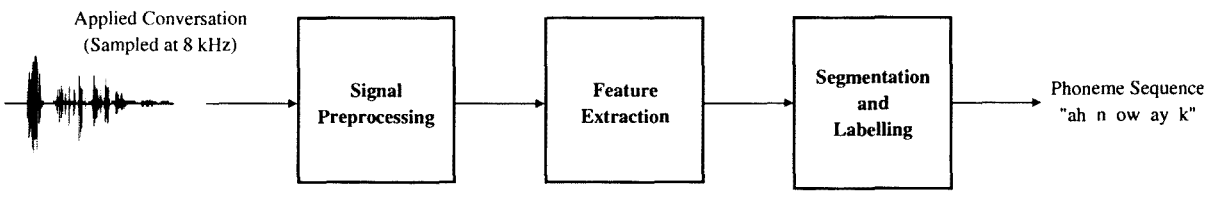


Figure 5.1: *Block diagram of the phoneme recognition front end.*

A description of each component follows below.

5.2.1 Signal Preprocessing

The speech signal is first preemphasized with a preemphasis filter. The filter has the following transfer function:

$$H(z) = 1 - 0.98z^{-1}. \tag{5.1}$$

The preemphasis filter helps to eliminate the effect of the larynx and the lips on the speech signal by increasing the relative energy of the high-frequency spectrum [40]. It also suppresses the speech signal's DC offset which is typically an artefact of the recording process. The frequency response of the filter is shown in Figure 5.2.

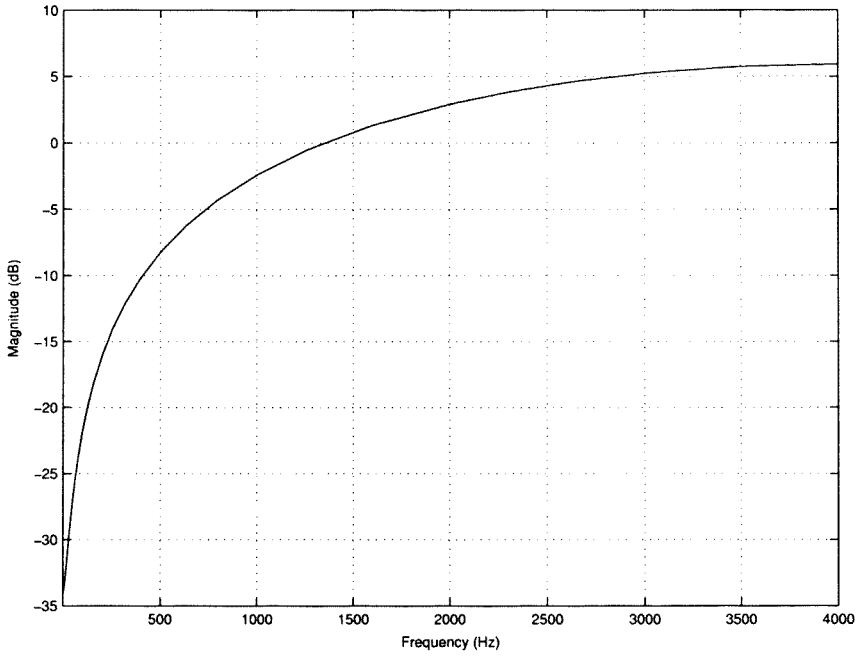


Figure 5.2: *Frequency response of the preemphasis filter.*

5.2.2 Feature Extraction

20-dimensional feature vectors are extracted over a fixed frame length of 20 ms every 10 ms, which is typical for frame based state-of-the-art speech recognition systems. The frame is obtained using the *Hamming* window function [41]. Assuming that the speech signal is quasi-stationary over the length of the analysis frame (a reasonable assumption since the frame is quite short), 10 linear predictive cepstral[1] coefficients (LPCCs) [1] together with their deltas[2] are calculated to form a 20-dimensional feature vector (this multi-dimensional feature vector will be referred to as a LPCC feature vector in this report). Note, however, that the zeroth cepstral coefficient is replaced with the frame energy before the delta features are determined. Cepstral mean subtraction [44] is also done beforehand in order to combat the effect of variability between different speakers and communication channels. Finally, the LPCC feature vector is normalised to prevent numerical underflow when using

[1]The word *cepstral* was coined as a word play on *spectral* [42].
[2]Delta features are used since they improve recognition accuracy [43].

covariance matrices in calculations.

5.2.3 Segmentation and Labelling

A phoneme spotter uses these extracted LPCC feature vectors to automatically segment and label the speech utterance into a string of phonemes. It is nothing more than a super-HMM structure into which pre-trained phoneme-HMMs (serving as models for the different phonemes and garbage classes) are embedded.

5.3 1996 ICSI Switchboard Phonetic Transcriptions

Since evaluation of the topic spotting systems had to be done on the Switchboard Corpus, hand-transcribed *phonemic* transcriptions of a portion of this database were needed for the training and testing of the phoneme recognition front end. Consequently, the 1996 ICSI (International Computer Science Institute) Switchboard *phonetic* transcriptions [18, 45] were employed for this purpose. The phonetic transcriptions were encoded with a variant of the *ARPAbet* transcription system [1] that was used for the TIMIT Acoustic-Phonetic Continuous Speech Corpus [46]. The *ARPAbet* transcription system was augmented with a set of *diacritics* representing such phonetic properties as glottalisation ("creaky voice"), nasalisation (typically applied to vocalic segments), frication, aspiration, de-voicing, unusual voicing and velarisation. In addition to this, transitional elements between adjacent vocalic or glide-like segments were explicitly marked.

The files associated with the 1996 ICSI Switchboard phonetic transcriptions are divided into the following four sets:

- **trn-chunk1.1**: Contains 746 speech segments corresponding to a total of 26.35 minutes of speech.

- **trn-chunk1.2**: Contains 253 speech segments corresponding to a total of 10.06 minutes of speech.
- **sri-align.1**: Contains 291 speech segments corresponding to a total of 12.6 minutes of speech.
- **dev-test**: Contains 451 speech segments corresponding to a total of 22.98 minutes of speech.

The telephone speech segments are given in *NIST/SPHERE* format and contain 16 bit data sampled at 8 kHz. Each segment contains the speech of only one speaker. The corresponding hand-transcribed word- and phonetic-transcription files[3] are given in *ESPS'* "xlabel"-format.

5.3.1 Merging the Transcription Files

Each speech segment's time-aligned transcription files (there is a word- and a phonetic-transcription file for each segment of speech) had to be merged, since the implemented diacritic stripper (Section 5.3.2) needed to know where the word boundaries lay with respect to the phonetic boundaries in order to remove the diacritics from the phonetic transcriptions. Consequently, a *Perl 5.0* script was written to accomplish this task. During the merging process, each word boundary was forced to correspond to the phonetic boundary closest to it. As a result, each amalgamated transcription file contained only phonetic boundaries with the word boundaries realigned to correspond to them.

5.3.2 Diacritic Stripping of the Phonetic Transcriptions

From the previous discussion it is clear that diacritic symbols were used to augment the standard phonemes to show phonetic detail. For the purposes of this research, pure phone-

[3]These files contain time-aligned transcriptions.

mic labels were needed though. A diacritic stripper was therefore implemented as a *Perl 5.0* script in order to fold the original transcriptions down to a base group of 38 phonemes and two garbage classes. The garbage classes were explicitly modelled as separate phonemes and consist of a general garbage class (representing non-speech sounds and silence) and a garbage class for stop consonant closures and releases (representing stop consonant closures and releases occurring without the other)[4]. The fact that the speech segments were transcribed at the sub-phonemic level made the implementation of the diacritic stripper unusually difficult.

A step-wise description of the diacritic stripping process follows below. Each step was applied to a merged time-aligned transcription file as a whole before applying the next step. Afterwards, the resulting time-aligned phonemic transcription file was saved in *PATREC's* "t96"-format.

1. Three phones, each having the same base form and occurring next to each other within a word, are folded into their common base form. The cases that occur in the transcription files are well-behaved. No special rules are therefore needed when combining these phones.

2. Two phones, each having the same base form and occurring next to each other within a word, are folded either into:

   - their common base form, if the diacritic symbols # (indicates a truncated phonetic segment) and *?* (indicates uncertainty about the identity of a phonetic segment) do not occur in the pair's diacritics, or
   - the general garbage class ***, if one or both of the diacritic symbols # and *?* occur in the pair's diacritics.

[4]Within the 1996 ICSI Switchboard phonetic transcription system, stop consonants are partitioned into closure and release components. However, in the phonetic transcription files some closures appear without their releases following them and some releases appear without their closures preceding them. For these instances, no meaningful stop consonants could thus be constructed during the diacritic stripping process. Despite this fact, it was felt that semantic meaning could still be attached to these specific closures and releases, rather than allocating them to the general garbage class. As a result, they were assigned to the newly created garbage class for stop consonant closures and releases.

Note that at least one of the phones in the pair should have diacritics in order for them to be folded into their common base form in accordance with the two points stated above. Under no circumstances will labels having the base forms *?* and *h#* be combined.

3. Phones having the following base forms or diacritics are folded into the general garbage class ***:
   - all non-speech events having the base form *h#*,
   - all unknown speech sounds having the base form *?*,
   - all glottal stops having the base form *q*, and
   - all phones containing the diacritics *#* and/or *?*.
4. Remove all the remaining diacritics.
5. Closures which are not followed by their respective releases are folded into the garbage class for stop consonant closures and releases *st**.
6. Releases which are not preceded by their respective closures are folded into the garbage class for stop consonant closures and releases *st**.
7. The closures that now remain are folded into their succeeding release base forms.
8. The following five allophones are folded into corresponding phonemes:
   - *ux* is folded into *uw*,
   - *axr* is folded into *er*,
   - *em* is folded into *m*,
   - *nx* is folded into *n*, and
   - *eng* is folded into *ng*.

9. There are six phoneme groups where within-group confusions are not counted:

   - *el* is folded into *l*,
   - *en* is folded into *n*,
   - *zh* is folded into *sh*,
   - *aa* is folded into *ao*,
   - *ix* is folded into *ih*, and
   - *ax* is folded into *ah*.

Table 5.1 shows the complete Switchboard 40-phoneme set that was used during this research. The phoneme occurrences are given for the 1996 ICSI Switchboard *phonemic* transcriptions as a whole. The resulting phoneme set closely resembles the one proposed by K. F. Lee and H. W. Hon [47].

No.	ID	Broad Phonemic Class	Example	Occurrences
1	***	GGC		5232
2	ih	VOW	bit	3340
3	ah	VOW	but	3110
4	n	NAS	non	2914
5	s	FRI	sack	1883
6	st*	GCR		1618
7	iy	VOW	beat	1610
8	eh	VOW	bet	1406
9	r	SVOW	red	1338
10	t	STP	tot	1260
11	ao	VOW	bought	1237
12	l	SVOW	led	1222
13	m	NAS	mom	1166
14	ae	VOW	bat	1151
15	dh	FRI	they	1057
16	ow	VOW	boat	1037
17	ay	VOW	bite	1033
18	k	STP	kick	1028
19	w	SVOW	wet	979
20	er	VOW	bird	921
21	z	FRI	zoo	871
22	uw	VOW	boot	689
23	y	SVOW	yet	685
24	v	FRI	very	675
25	d	STP	dad	663
26	ey	VOW	bait	650
27	p	STP	pop	645
28	b	STP	bob	599
29	f	FRI	fin	596
30	dx	FT	muddy	580
31	uh	VOW	book	576
32	hh	FRI	hay	457
33	ng	NAS	sing	411
34	th	FRI	thief	331
35	g	STP	gag	325
36	sh	FRI	shoe	305
37	aw	VOW	bough	290
38	jh	AFR	judge	202
39	ch	AFR	church	181
40	oy	VOW	boy	41

Table 5.1: *The Switchboard 40-phoneme set (GGC = general garbage class; GCR = garbage class for stop consonant closures and releases; VOW = vowel; SVOW = semi-vowel; NAS = nasal; FRI = fricative; STP = stop consonant; AFR = affricate; FT = flaps and trills)*

5.4 Training of the Phoneme-HMMs

Standard three-state left-to-right one-skip HMMs, using 20-dimensional *full* covariance Gaussian pdfs for the emission states, served as context independent models[5] for the different phonemes (this includes the two garbage classes that were explicitly modelled as separate phonemes). The phoneme-HMMs were trained on the *phonemic* transcriptions of the *trn-chunk1.1*, *trn-chunk1.2* and *sri-align.1* datasets[6] (approximately 49 minutes of speech in total):

- **Initialisation**: The first step was to initialise the hidden Markov models:
  - The transition probabilities for each phoneme-HMM were set as depicted in Figure 5.3. These initial transition probabilities were chosen, since they ensured fast convergence during training.

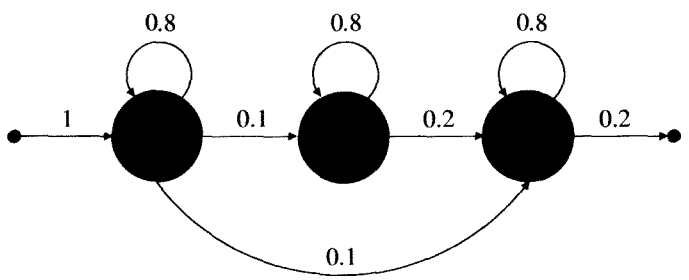


Figure 5.3: *Initialisation of a phoneme-HMM.*

  - The LPCC feature vectors for each observation of a certain phoneme class were equally split over the three states of the corresponding phoneme-HMM.
  - The probability density functions were initialised to a mean of zero and *unit* covariance matrix.

[5]Context independent models were used instead of context dependent ones, because of data scarcity.

[6]The following two speech segments that occur in both the *trn-chunk1.1* and *sri-align.1* datasets, were excluded from the latter dataset for training purposes: 2780-A-0005 and 2830-A-0020. Note that they are the only speech segments that overlap in the 1996 ICSI Switchboard phonetic transcriptions.

- One iteration of Viterbi training [48] was done in order to increase the convergence speed of the subsequent training of the phoneme-HMMs and to reduce the effect of non-optimal local maxima.

- **Training**: The phoneme-HMMs were then trained using the Viterbi reestimation algorithm which is based on the expectation maximisation (EM) principle [49]. Consequently, the reestimation process continued until the improvement, I, in the total training score (sum of log-likelihoods for all the observations over all the different phoneme-HMMs):

$$I = \frac{old\ score - new\ score}{old\ score}, \tag{5.2}$$

fell below the predefined threshold of 5×10^{-4}. The total training score as a function of the Viterbi iteration number is shown in Figure 5.4.

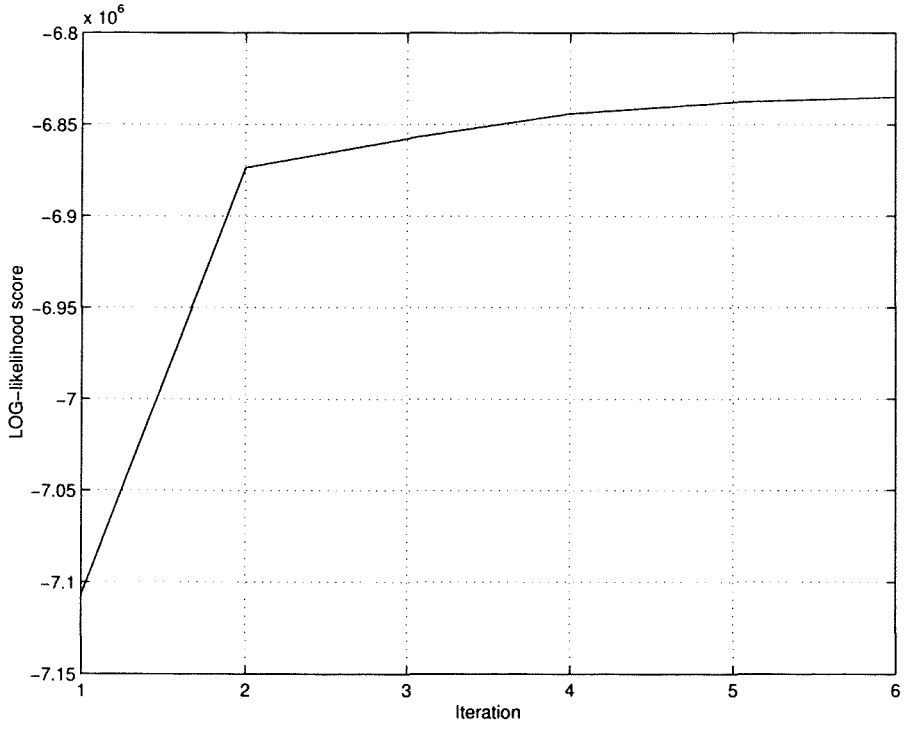


Figure 5.4: *Training score as a function of the Viterbi iteration number.*

5.5 Phoneme Spotter

A phoneme spotter was subsequently implemented and is nothing more than a super-HMM structure consisting of the trained phoneme-HMMs. The phoneme spotter was employed to automatically segment and label a speech utterance, represented by a 20-dimensional stream of LPCC feature vectors, into a string of phonemes using a maximum likelihood approach. This was accomplished with the help of the Viterbi algorithm by determining the optimal path through the spotter-HMM.

A block diagram of the phoneme spotter is depicted in Figure 5.5 (adapted from Figure 3.13 in [50]). From the figure it is clear that the *a priori* probabilities of the different phonemes were also embedded in the structure by setting the transition probabilities from the entry null state of the super-HMM structure to the first emission state of each phoneme-HMM equal to the *a priori* probability of that phoneme. $P(q_k)$ therefore represents the *a priori* probability of phoneme k in the figure. The *a priori* probabilities were estimated by simply calculating the occurrence frequency of each phoneme in the *trn-chunk1.1*, *trn-chunk1.2* and *sri-align.1* datasets, the same datasets that were used during the training of the phoneme-HMMs. Figure 5.6 shows the *a priori* distribution of the phonemes in the front end's training set.

An example of a time-aligned phonemic transcription that was generated by the phoneme recogniser is shown in Table 5.2. It is of speech segment 2151-A-0009 that was obtained from the *dev-test* dataset. Its true phoneme sequence can be seen in Table 5.3. Both tables are given in *PATREC's* "t96"-format. It is apparent from these tables that the front end failed to generate the correct phoneme sequence for the speech segment.

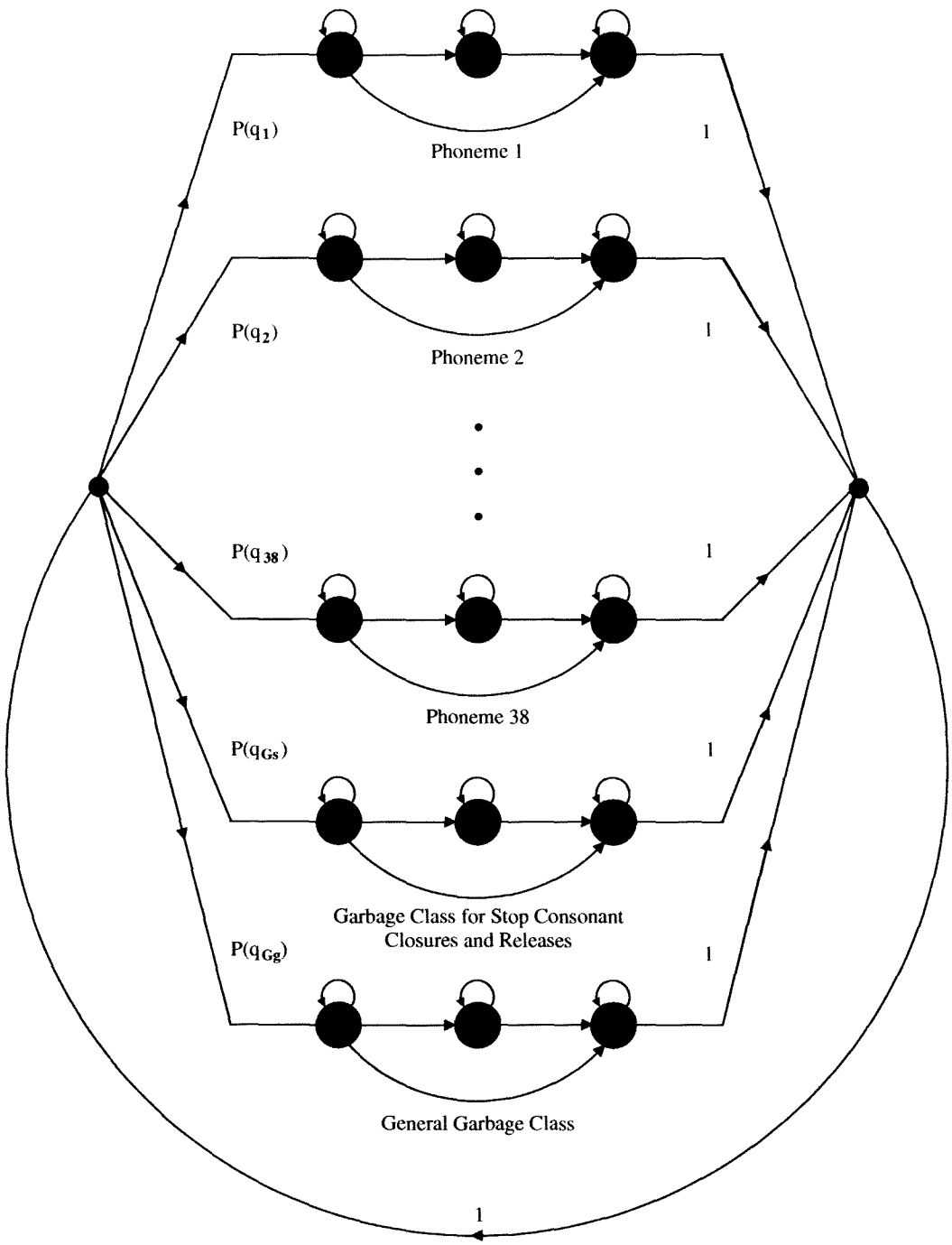


Figure 5.5: *Block diagram of the phoneme spotter.*

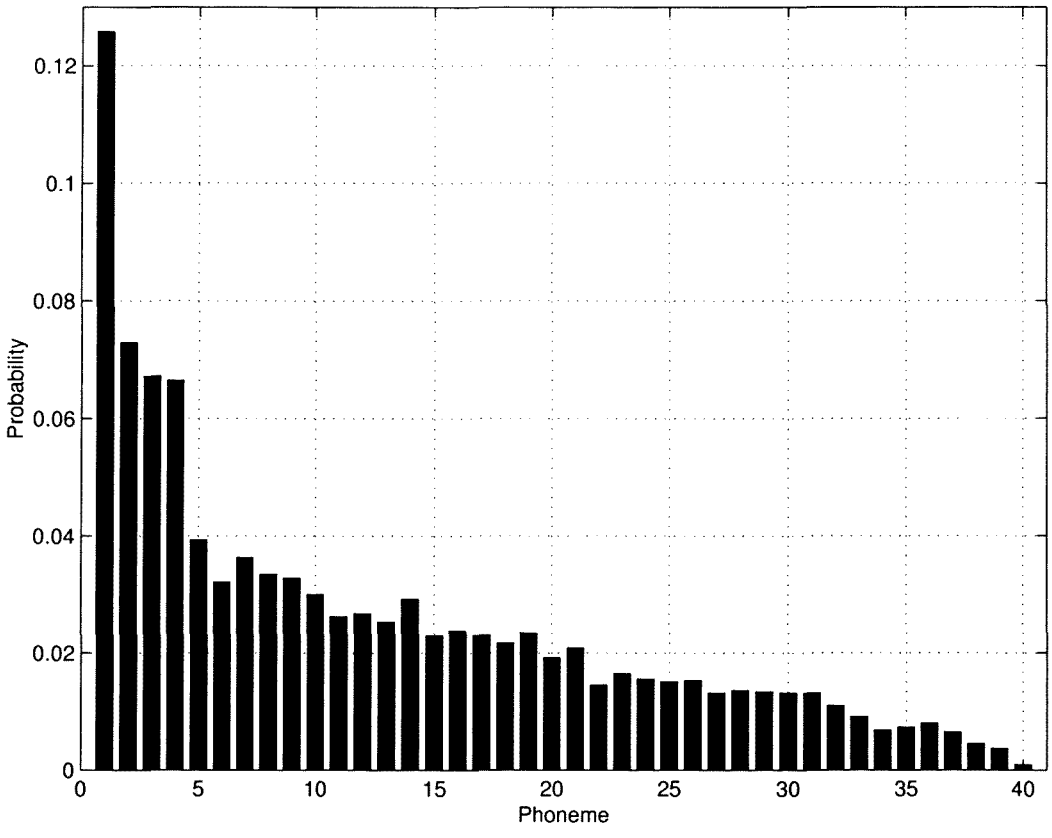


Figure 5.6: *A priori distribution of phonemes in the front end's training set.*

Time (s)	Phoneme
0.06	th
0.11	ow
0.23	l
0.34	p
0.41	f
0.55	ao
0.69	l
0.86	uh
0.95	dx
1.30	l
1.36	b
1.42	uw
1.46	m
1.56	l
1.61	b
1.71	uh
1.80	p
2.00	er
2.09	f
2.20	uw
2.28	s
2.42	***
2.60	ow
2.64	jh
2.74	s
2.85	w
2.95	ao
3.00	dx

Table 5.2: *Front end's time-aligned phonemic transcription of speech segment 2151-A-0009.*

Time (s)	Phone	Phoneme	Word
0.05057300	h#	***	-
0.06567800	tcl	st*	-
0.24315900	ow_n	ow	don't
0.32120000	q	***	-
0.41812300	f	f	-
0.55532500	aa	ao	-
0.59938000	l_dl	l	-
0.67742200	ow	ow	follow
0.74665200	uh	uh	-
0.76679100	dx_t	dx	it
0.86000000	ix	ih	-
0.94175500	d_fr_!	st*	at
1.20743100	ao	ao	-
1.32449300	el	l	all
1.34966700	dcl	-	-
1.36603100	d	d	do
1.38239400	y	y	-
1.42645000	ix	ih	you
1.45036600	n	n	-
1.56994500	ow_n	ow	know
1.60770700	dh	dh	-
1.66309100	eh	eh	-
1.72980400	?	***	there
1.81036300	f	f	-
1.91483800	er	er	-
2.00000000	iy	iy	very
2.10206500	f	f	-
2.12724000	y	y	-
2.20905800	uw	uw	few
2.30000000	s	s	-
2.33000000	kcl	-	-
2.38000000	k	k	-
2.56779600	aw	aw	-
2.60000000	tcl	st*	-
2.67730600	s	s	scouts
2.76580300	g	st*	-
2.82999800	ow	ow	go
2.95461300	aa_n	ao	-
2.98356400	nx	n	on

Table 5.3: *Time-aligned hand-transcriptions of speech segment 2151-A-0009.*

5.6 Evaluation of the Phoneme Recognition Front End

A phoneme classifier was implemented in order to evaluate the performance of the phoneme recognition front end on the phonemes obtained from the diacritically stripped *dev-test* dataset (approximately 23 minutes of speech). Insertion- and deletion-errors were not taken into account during the evaluation process. To classify a phoneme, Bayes' *maximum a posteriori* (MAP) decision rule [51] was employed:

$$k = \arg\max_k P(q_k|\mathbf{z}), \tag{5.3}$$

where $P(q_k|\mathbf{z})$ is the *a posteriori* probability of q_k given the phoneme's acoustic observation sequence $\mathbf{z}$ (consisting of LPCC feature vectors). Its value can be determined using Bayesian inversion:

$$P(q_k|\mathbf{z}) = \frac{f(\mathbf{z}|q_k)P(q_k)}{f(\mathbf{z})}, \tag{5.4}$$

where $f(\mathbf{z}|q_k)$ is the value of the probability density function which is calculated by the phoneme-HMM of q_k for $\mathbf{z}$. $P(q_k)$ is the *a priori* probability of phoneme k. The normalising pdf $f(\mathbf{z})$ is the same for all the phoneme classes and thus causes an arbitrary, but *common* scaling of the *a posteriori* probabilities. As a result, Equation 5.3 can be written as:

$$\begin{aligned} k &= \arg\max_k f(\mathbf{z})P(q_k|\mathbf{z}) \\ &= \arg\max_k f(\mathbf{z}|q_k)P(q_k), \end{aligned} \tag{5.5}$$

without affecting the outcome of the MAP decision rule. The phoneme classifier was therefore implemented as shown in Figure 5.7 (adapted from Figure 3.12 in [50]).

The recognition accuracy for each individual phoneme was subsequently calculated as follows:

$$R_k = \frac{number\ of\ correct\ classifications\ of\ phoneme\ k}{total\ number\ of\ occurrences\ of\ phoneme\ k}, \tag{5.6}$$

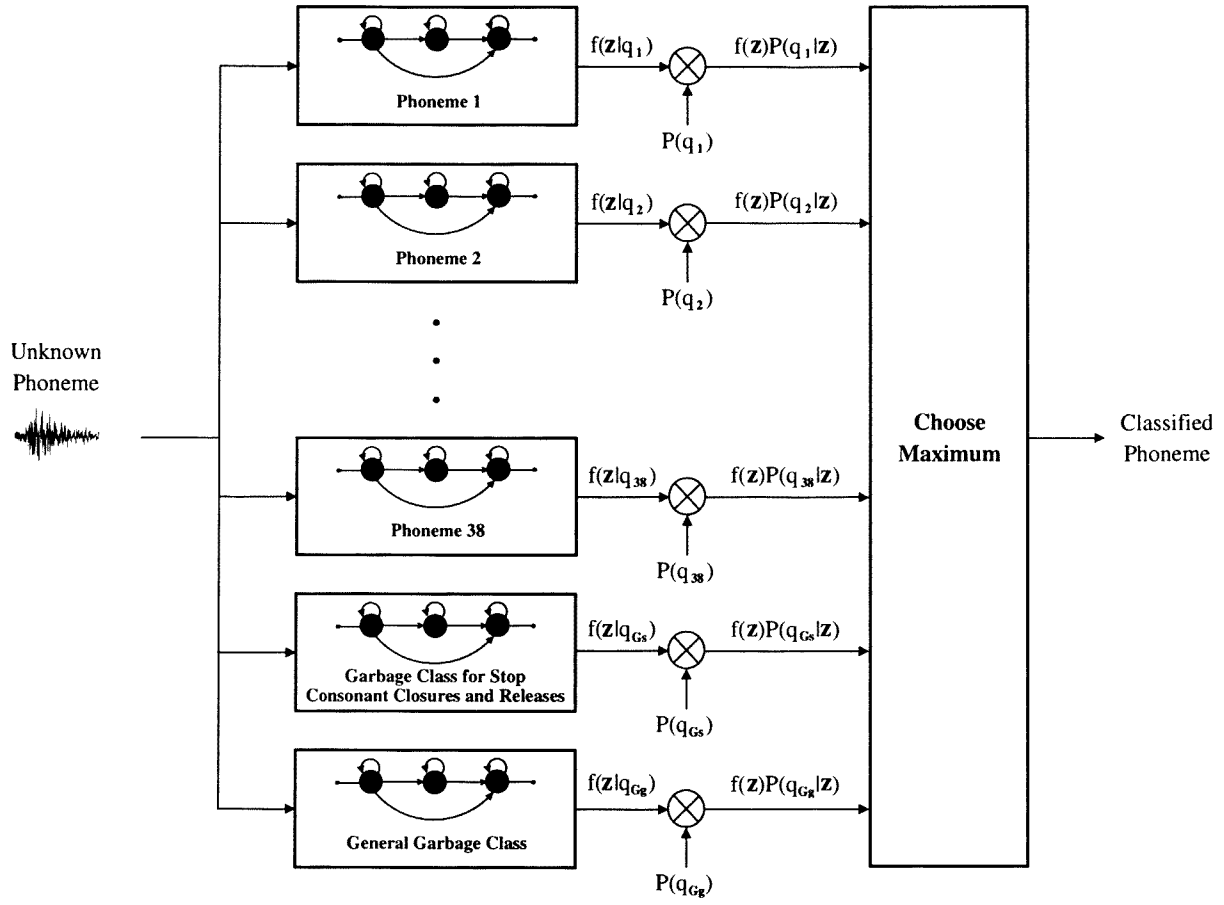


Figure 5.7: *Block diagram of the phoneme classifier.*

where R_k is the recognition accuracy for phoneme k. The recognition rates are presented in Figure 5.8. The recognition accuracy of the front end, R, was then calculated as shown below:

$$R = \sum_k P(q_k) R_k. \tag{5.7}$$

An overall recognition accuracy of 43.1% was consequently measured. This is indicated by the horizontal line in Figure 5.8.

A context independent confusion matrix was produced during the evaluation process. Its 3D confusion plot is illustrated in Figure 5.9. Correct classifications are located on the main diagonal, while incorrect classifications ("confusions") lie off the main diagonal. Note that this confusion matrix was employed by those topic spotters that generated *soft* counts for their lexicon members.

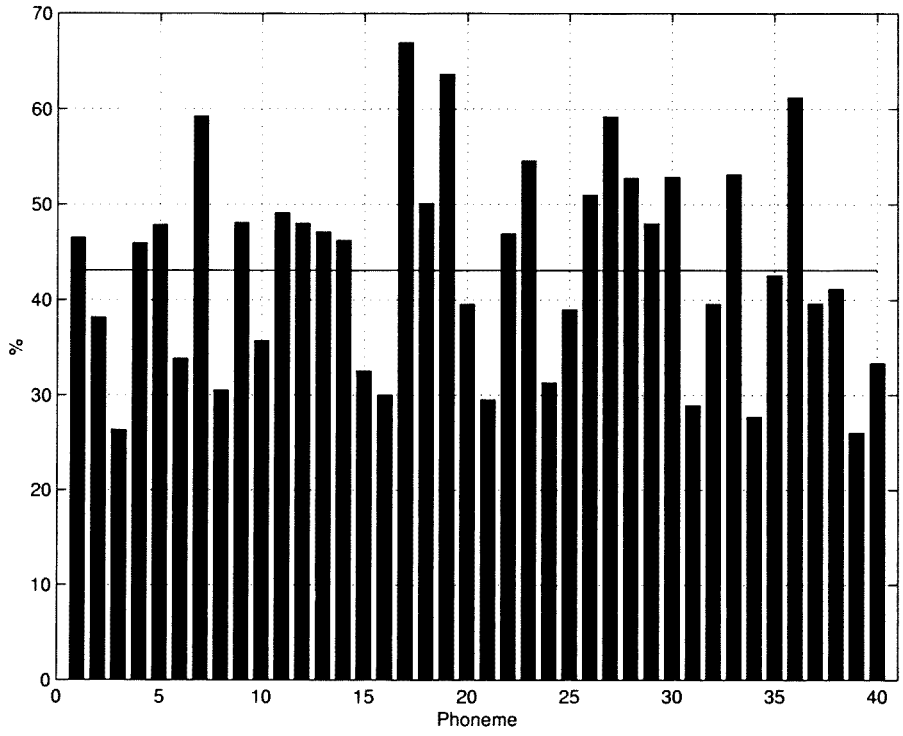


Figure 5.8: *Recognition accuracy for individual phonemes. The overall recognition accuracy of the phoneme recognition front end is indicated by the horizontal line.*

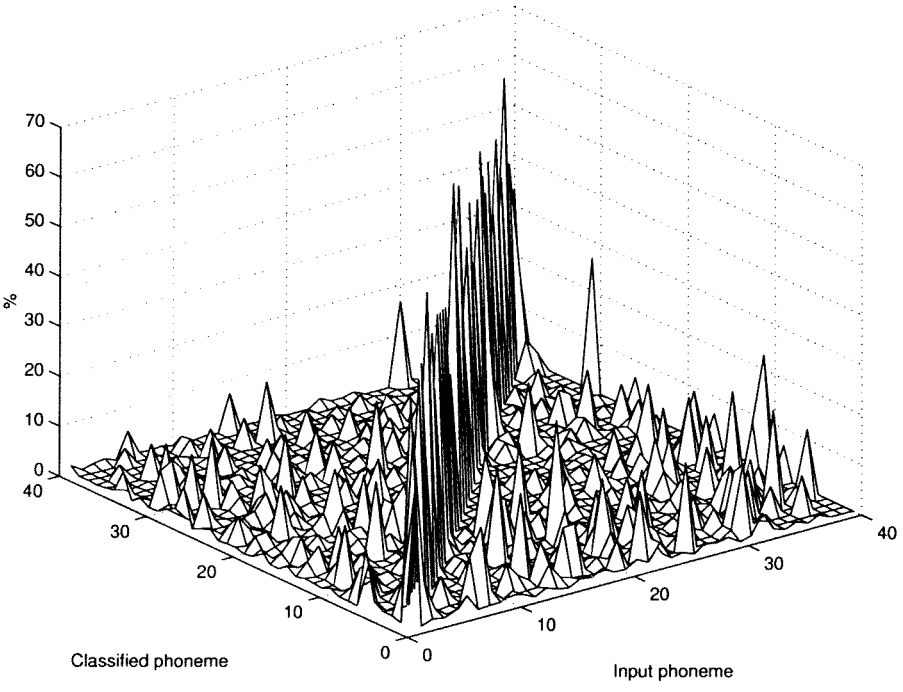


Figure 5.9: *3-dimensional confusion plot of phoneme recognition experiment.*

5.7 Summary

This chapter described how signal processing-, feature extraction-, pattern recognition-, and statistical modelling-techniques were used to realise a practical, speaker independent, context independent phoneme recogniser with an overall recognition accuracy of 43.1%. It was subsequently employed to phonemically transcribe CRIM's subset of the Switchboard Corpus (Section 6.2.1) in order to allow comparison of the different topic spotting algorithms with each other.

Chapter 6

Experiments and Results

I venture to define science as a series of interconnected concepts and conceptual schemes arising from experiment and observation and fruitful of further experiments and observations. The test of a scientific theory is, I suggest, its fruitfulness.

— James B. Conant

6.1 Introduction

The purpose of this chapter is to report on the experimental setup, hardware specifications, software implementation, and most important experiments. No results are presented for those adjustments that yielded a deterioration in topic spotting performance.

The experimental setup is discussed in Section 6.2. Details about CRIM's subset of the Switchboard Corpus are presented, followed by an explanation of how the performance of each topic spotter was evaluated. Afterwards, the test for statistical significance is closely examined. An overview is then given of the experiments that were conducted. Next, the hardware specifications are presented in Section 6.3. This is followed by an explanation in Section 6.4 of how the software was implemented. Finally, Section 6.5 reports on the

results of the most important experiments.

6.2 Experimental Setup

6.2.1 Switchboard Corpus

The Switchboard Corpus [35], collected at Texas Instruments, contains 2430 conversations (4860 channels[1]) covering 52 topics and comprising 240 hours of recorded speech. Over 500 speakers of both sexes from every major dialect of American English participated in the project. It was specifically designed to support the development of telephone-based speech technology as well as basic research on spontaneous conversational speech and language.

CRIM's subset of the Switchboard Corpus was used during this research for experimental purposes, and consists of 506 channels covering 10 topics. The channels of interest were extracted from the corpus with the help of the *PATREC* software system as follows:

1. The 345 conversations, containing the 506 channels of interest, were split into their respective channels.
2. Each conversation's channels were then compared with one another in order to remove silence and reduce/eliminate the effects of cross talk:
    - If a speech segment from the one channel had more than twice the energy of the time-aligned speech segment from the other channel, the speech segment with the highest energy was kept, while the other one was removed.
    - If a speech segment from the one channel had less than twice the energy of the time-aligned speech segment from the other channel, both speech segments were removed from their respective channels.

[1]Each conversation contains two channels, where a channel represents the speech of only one speaker.

- If the speaker was active for less than 400 ms in a speech segment, it was removed.
- Speech segments containing silence of shorter than 100 ms were not removed. This was done in order to protect closures.

3. Finally, the 506 channels of interest (approximately 1462 minutes of speech) were extracted.

These channels were subsequently arranged to form a training- and a testing-set in the proportion 9:1. The training set was also split up to form a training- and a validation-subset in the proportion 4:1, with these subsets used in conjunction with the extended training criterion (Section 2.2.5) to estimate the appropriate size of the final lexicon as a percentage.

The topic distribution of the 506 channels can be seen in Table 6.1. From the table it is obvious that the topics are such that the channels may well fit more than one topic (e.g. "crime" and "gun control"). However, since each channel is allocated to only one topic in the corpus, many detections that registered as false alarms may in fact be correct.

Topic (10)	**Topic Number**	**Number of Channels (506)**	**Training Set (455)**	**Testing Set (51)**	**Training Subset (360)**	**Validation Subset (95)**
Air Pollution	302	36	32	4	25	7
Music	308	51	46	5	36	10
Crime	312	53	48	5	38	10
Gun Control	314	38	34	4	27	7
Buying a Car	320	59	53	6	42	11
Universal Public Service	327	27	24	3	19	5
Pets	351	69	62	7	49	13
Public Education	353	60	54	6	43	11
Exercise and Fitness	356	55	50	5	40	10
Family Life	358	58	52	6	41	11

Table 6.1: *Topic distribution of the 506 channels.*

It was determined that 0.23% of the topic spotting data was used during the training of the phoneme recognition front end. Since the overlap was small the marginal effect that it had on results was ignored.

6.2.2 Method of Evaluation

The task was a detection problem. System evaluation was therefore performed by means of an ROC curve. The error area above it was used to evaluate system performance.

6.2.3 Statistical Significance

Statistical significance of the results, represented by the ROC curves of any two topic spotting systems, was verified by means of the McNemar test [32, 33] modified (Section 6.2.3.1) to compensate for the fact that the task was one of detection rather than classification. Since an algorithm's detection errors consist of missed detections and false alarms, it was decided to perform separate McNemar tests over a range of error indices, where each index represents a specific missed detection rate to false alarm rate.[2] A two-tailed probability was therefore calculated for each error index in order to determine whether the measured difference between the topic spotting systems could have arisen under the null hypothesis[3].

During this research, separate McNemar tests were performed over a range of error indices spanning from 0.1 to 10, with the threshold (probability) for rejecting the null hypothesis set to 0.1[4]. The null hypothesis was consequently rejected for those McNemar probabilities that fell below this threshold value. Note that the tests for statistical significance were limited to these error indices, because the ROC curves are usually very close to one another for indices outside of this range.

[2]The topic spotters will usually have different decision thresholds for the same error index.
[3]The null hypothesis states that both algorithms are equally likely to make *detection* errors.
[4]In the field of speech recognition it is standard practise to select this rejection threshold (α).

In this chapter, each ROC graph of two competing topic spotting systems is accompanied by a graph showing the outcome of the McNemar tests. The graphs will be depicted as shown in Figures 6.1 and 6.2. Note that:

- Statistical significance of the results was verified for those ROC segments falling inside the shaded region of the ROC graph:
  - The line at the top of the shaded region represents an error index of 0.1.
  - The line extending from the top left-hand side to the bottom right-hand side of the ROC graph represents an error index of 1.
  - The line at the left-hand side of the shaded region represents an error index of 10.
- The threshold for rejecting the null hypothesis will be indicated by a dotted line on the McNemar graph.

Several *Matlab* functions were written in order to accomplish the task described above.

6.2.3.1 Modified McNemar Test

The McNemar test can be used for classification problems in order to determine whether the measured difference in performance between two algorithms on the same dataset is statistically significant. This is accomplished by comparing the classification decisions of the algorithms, which are assumed to be statistically independent, with one another. The comparison is performed by postulating the null hypothesis which states that the algorithms are equally likely to make *classification* errors. A two-tailed probability is then calculated to determine whether the null hypothesis is valid.

As was mentioned previously, the task during this research was one of detection rather than classification. The null hypothesis was therefore redefined to state that the two algorithms

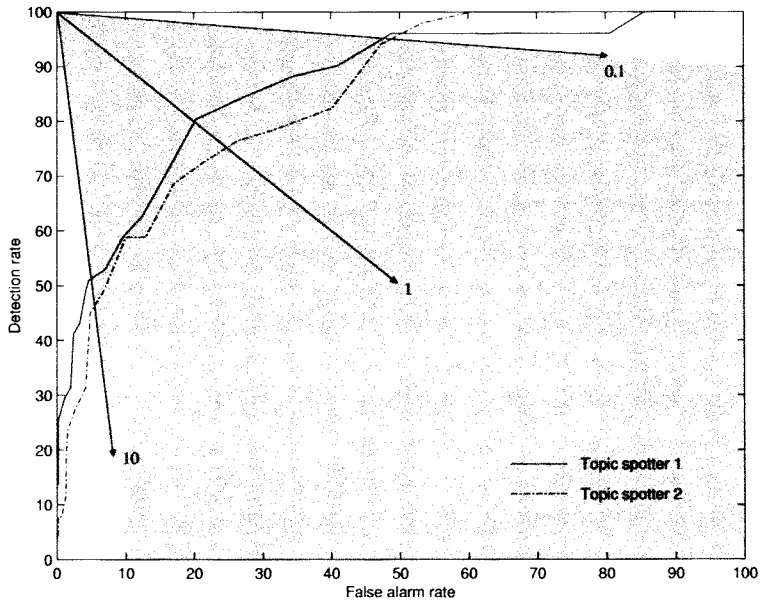


Figure 6.1: *ROC graph: Topic spotter 1 vs. Topic spotter 2. The line at the top of the shaded region represents an error index (missed detection rate:false alarm rate) of 0.1, the line extending from the top left-hand side to the bottom right-hand side of the ROC graph an index of 1, and the line at the left-hand side of the shaded region an index of 10.*

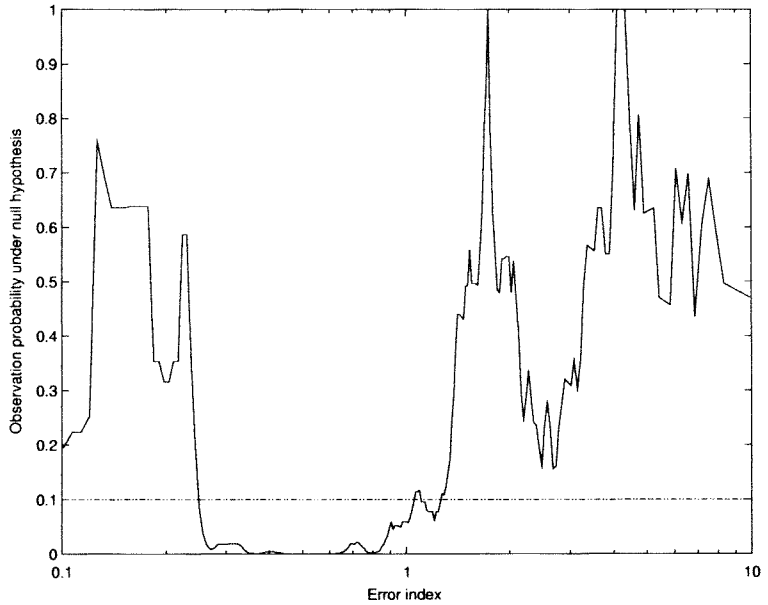


Figure 6.2: *McNemar graph: Topic spotter 1 vs. Topic spotter 2. The threshold for rejecting the null hypothesis is indicated by the dotted line.*

are equally likely to make *detection* errors (sum of missed detections and false alarms). Consequently, the McNemar test had to be modified.

Let T_1 and T_2 represent topic spotter 1 and topic spotter 2 respectively. The joint performance of the two algorithms on the same dataset can then be summarised as shown in Table 6.2.

		T_2	
		Correct	Incorrect
T_1	Correct	n_{00}	n_{01}
	Incorrect	n_{10}	n_{11}

Table 6.2: *Joint performance of the two topic spotters on the same dataset.*

In this table:

- n_{00} = the number of times that both algorithms made the correct detection decision[5].
- n_{01} = the number of times that T_1 made the correct detection decision, while T_2 did not.
- n_{10} = the number of times that T_2 made the correct detection decision, while T_1 did not.
- n_{11} = the number of times that both algorithms made the incorrect detection decision.

In a similar manner the joint probabilities ϕ_{00}, ϕ_{01}, ϕ_{10} and ϕ_{11} can be defined. For example:

$$\begin{aligned} \phi_{00} &= P(T_1 \textit{ does not make an error}, T_2 \textit{ does not make an error}) \\ &= \frac{n_{00}}{n}, \end{aligned} \tag{6.1}$$

where n equals the total number of detection decisions ($n_{00} + n_{01} + n_{10} + n_{11}$).

[5]The statistically independent binary decision whether or not to allocate a topic to a conversation.

The null hypothesis dictates that both algorithms are equally likely to make detection errors. Consequently, this implies that their error rates should be equal:

$$e_1 = e_2, \tag{6.2}$$

where $e_1 = \phi_{10} + \phi_{11}$ (T_1's error rate) and $e_2 = \phi_{01} + \phi_{11}$ (T_2's error rate). Equation 6.2 can therefore be written as follows:

$$\begin{aligned} \phi_{10} + \phi_{11} &= \phi_{01} + \phi_{11} \\ \therefore\ \phi_{10} &= \phi_{01}. \end{aligned} \tag{6.3}$$

The conditional probability ϕ_{T_1}, that T_1 makes an incorrect detection decision given that only one of the two algorithms makes an error, should consequently be equal to 0.5:

$$\begin{aligned} \phi_{T_1} &= \frac{\phi_{10}}{\phi_{01} + \phi_{10}} \\ &= 0.5. \end{aligned} \tag{6.4}$$

As a result, given that only one of the algorithms makes an incorrect detection decision, it is equally likely to be either one. To test the null hypothesis it is therefore sufficient to examine the detection decisions on which only one topic spotter makes an error ($n_{01} + n_{10}$).

If the test is thus conditioned on the number of detection decisions $k = n_{01} + n_{10}$, N_{10} (n_{10}'s random variable) should have a binomial distribution $\beta(k, \frac{1}{2})$ under the null hypothesis. The validity of the null-hypothesis is then determined by applying a two-tailed test to the observation of a random variable M drawn from $\beta(k, \frac{1}{2})$:

$$P_2 = 2P(n_{10} \leqslant M \leqslant k) \quad when\ n_{10} > \frac{k}{2} \tag{6.5}$$

$$= 2P(0 \leqslant M \leqslant n_{10}) \quad when\ n_{10} < \frac{k}{2} \tag{6.6}$$

$$= 1 \quad when\ n_{10} = \frac{k}{2}. \tag{6.7}$$

These two-tailed probabilities are calculated directly as follows:

$$P_2 = 2\sum_{m=n_{10}}^{k}\binom{k}{m}\left(\frac{1}{2}\right)^k \quad when\ n_{10} > \frac{k}{2} \tag{6.8}$$

$$= 2\sum_{m=0}^{n_{10}}\binom{k}{m}\left(\frac{1}{2}\right)^k \quad when\ n_{10} < \frac{k}{2}. \tag{6.9}$$

After determining P_2 for n_{10}, the null hypothesis is rejected if the two-tailed probability is below some significance level α (rejection threshold).

6.2.4 Experiments

Initial experiments were conducted to determine optimal parameters for lexicon initialisation. The simultaneous inclusion of 2- and 3-grams was investigated, as well as the simultaneous inclusion of 2-, 3- and 4-grams. In addition to this, recurrence thresholds of 2, 3 and 4 were evaluated. For the topic spotting systems of Section 6.5 (i.e. ENWN, SMART, and extended SMART), best performance was obtained when phoneme n-grams in the length range 2 to 4 were allowed, and when the recurrence threshold was set to 2 so that those keystrings that occurred more than once in the training data were included.[6] Consequently, the initial *untrained* lexicon of each topic spotter contained a total of 189451 keystrings (1555 2-gram, 43426 3-grams, 144470 4-grams). Note that the results presented in this chapter are given for these parameter values.

Closed-set[7] experiments were conducted to evaluate the topic spotting algorithms. This was done in order to achieve better utilisation of the available data. The results obtained for the most important experiments will be presented in Section 6.5.

[6]This result makes intuitive sense, since more keystrings were available for selection by the training algorithm of each topic spotter when these parameters were used.

[7]The testing data contains topics that are present in the training data.

6.3 Hardware

Experiments were carried out on an *Intel Pentium III Coppermine 600EB* system. The computer had 640 MB of RAM and *Redhat Linux 6.2* was installed.

6.4 Software

The topic spotting systems were implemented in the *PATREC* software system. Of particular importance was the implementation of ENWN and SMART. Scheffler's code had to be rewritten in order to allow efficient and practical comparison on the Switchboard Corpus. The following modifications were consequently introduced:

- Pointer code was used at certain critical stages of the program to reduce computational overhead when working with its data structures.
- Lookup tables were employed. For example, rather than searching sequentially through the lexicon to determine the presence of a keystring, a lookup table was used instead.
- SMART's sequence matchers were embedded into a tree structure to exploit similarities existing between them. Consider as an example the hypothetical lexicon (Figure 6.3) generated by SMART. Its sequence matchers are shown in Figure 6.4. Exploiting the similarities between these sequence matchers yields the tree structure in Figure 6.5. Fitting each observed keystring $\mathbf{y}$[8] to this structure results in a significant improvement in SMART's simulation time, since multiple calculations of the same values are avoided. For a full description of how to generate conversation- and topic-vectors using a tree structure, refer to the flow diagram in Figure A.17 (Appendix A).

[8] $\mathbf{y}$ should have the same length as the longest lexicon member(s) if it is going to be fitted to a tree structure.

```
z er
uw g th ao
er d jh
uw g
l m ao
uw g th z
z er ao uh
er d jh ch
uw g th iy
er d
z er uw z
l m ow
y v
z er ao k
uw g th sh
z er ao
uw g th
uw g th n
l m
z er uw
```

Figure 6.3: *The hypothetical lexicon generated by SMART.*

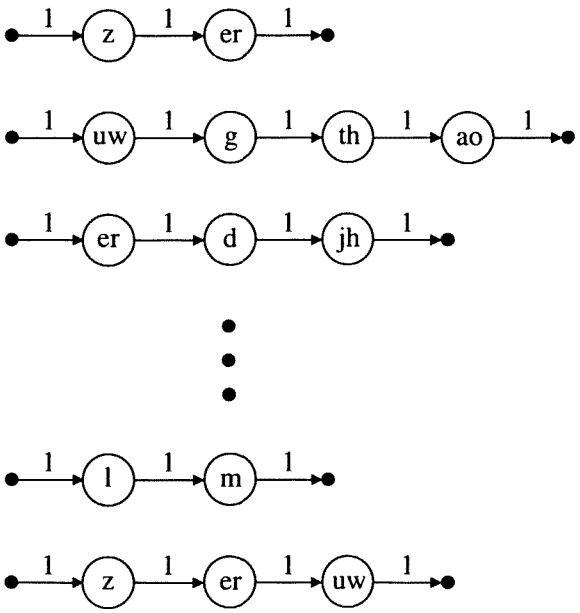


Figure 6.4: *Sequence matchers corresponding to keystrings in SMART's hypothetical lexicon.*

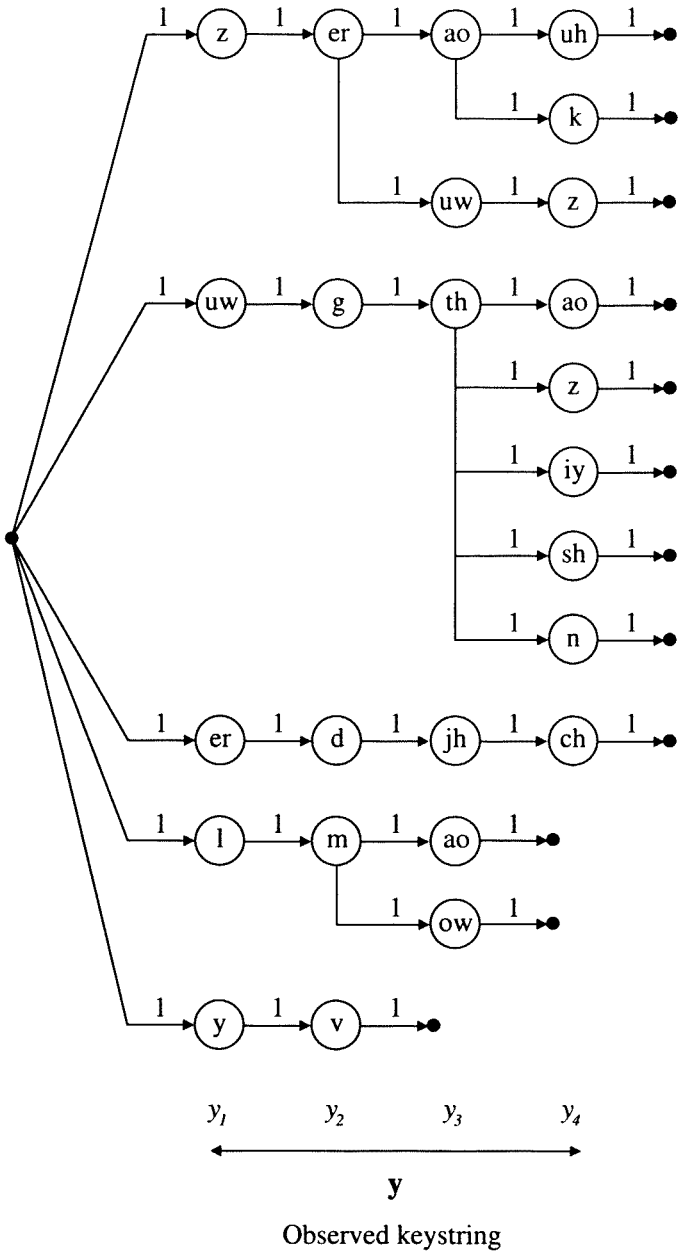


Figure 6.5: *Tree structure corresponding to the hypothetical lexicon's sequence matchers.*

- Where necessary, the layout of the code was changed.

Simulation times of ENWN and SMART were considerably reduced. This is clearly evident from Table 6.3. Note that conservative estimates are given for the simulation times of Scheffler's implementation of ENWN and SMART.

Topic Spotter	Simulation Time (days) Before	After	Percentage Reduction
ENWN	21.1	0.29	98.6%
SMART	96	1.82	98.1%

Table 6.3: *Simulation times of ENWN and SMART — phoneme n-grams in the length range 2 to 4 were allowed, and the recurrence threshold was set to 2.*

An added benefit of the optimisation process was a reduction in memory usage. Scheffler's implementation of ENWN and SMART uses more than 820 MB of RAM. The new implementation, on the other hand, uses less than 600 MB of RAM.

Flow diagrams of ENWN and SMART, before and after the modifications were introduced, are shown in Appendix A.

6.5 Results

6.5.1 ENWN versus SMART

In this section the performances of ENWN, SMART, and a system using soft counts along with the Euclidean distance measure are presented. The latter topic spotter will be referred to as the SE algorithm.

Before SMART was compared as a whole to ENWN, the introduction of each individual enhancement[9] by SMART was investigated:

- **Introducing soft counts — ENWN vs. SE**: The training curves of ENWN and SE are shown in Figures 6.6 and 6.7 respectively. Comparative results for these topic spotters are illustrated in Figure 6.8. The replacement of ENWN's hard counts with SMART's soft counting strategy yields an 11.5% reduction in ROC error area. The McNemar graph (Figure 6.9) indicates that regions of statistical significance exist.
- **Introducing the CE distance measure — SE vs. SMART**: The training curve of SMART is depicted in Figure 6.10. The ROC graph of SE versus SMART is shown in Figure 6.11. Replacing SE's Euclidean norm with the CE distance measure results in a reduction of 27.5% in ROC error area. The outcome of the McNemar tests is presented in Figure 6.12. It is apparent from this graph that the measured differences between SE and SMART are statistically significant over a large range of error indices.

When SMART is compared to ENWN, a 35.8% reduction in ROC error area is observed. Comparative results are shown in Figure 6.13. The McNemar graph of Figure 6.14 indicates that the improvement is statistically significant over virtually the entire range of error indices that were investigated.

[9]The enhancements introduced by SMART are the use of soft counts and the CE distance measure.

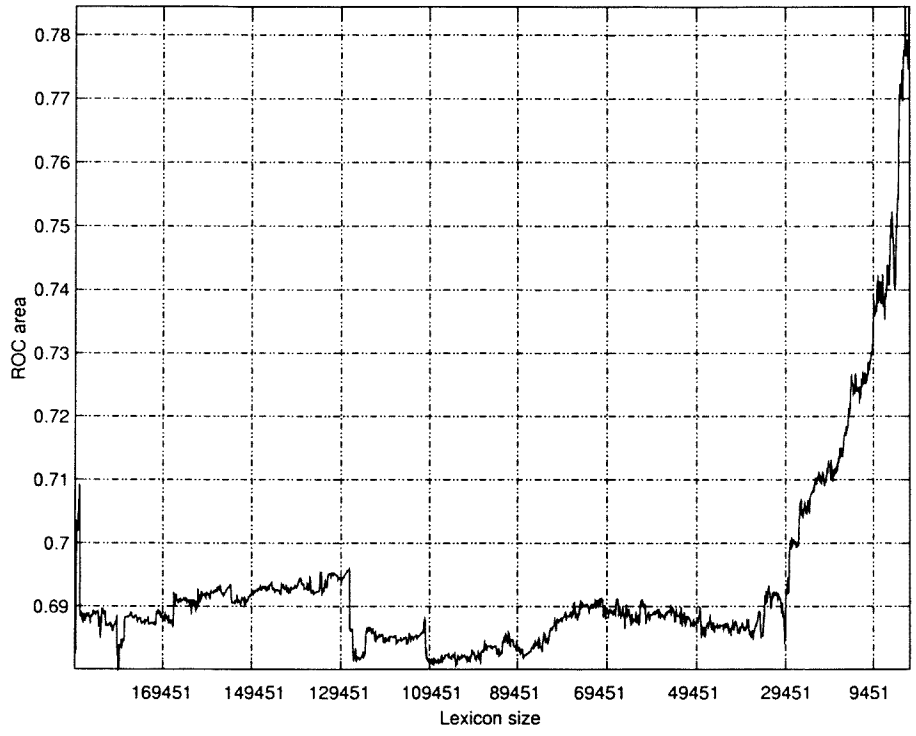


Figure 6.6: *ENWN's training curve.*

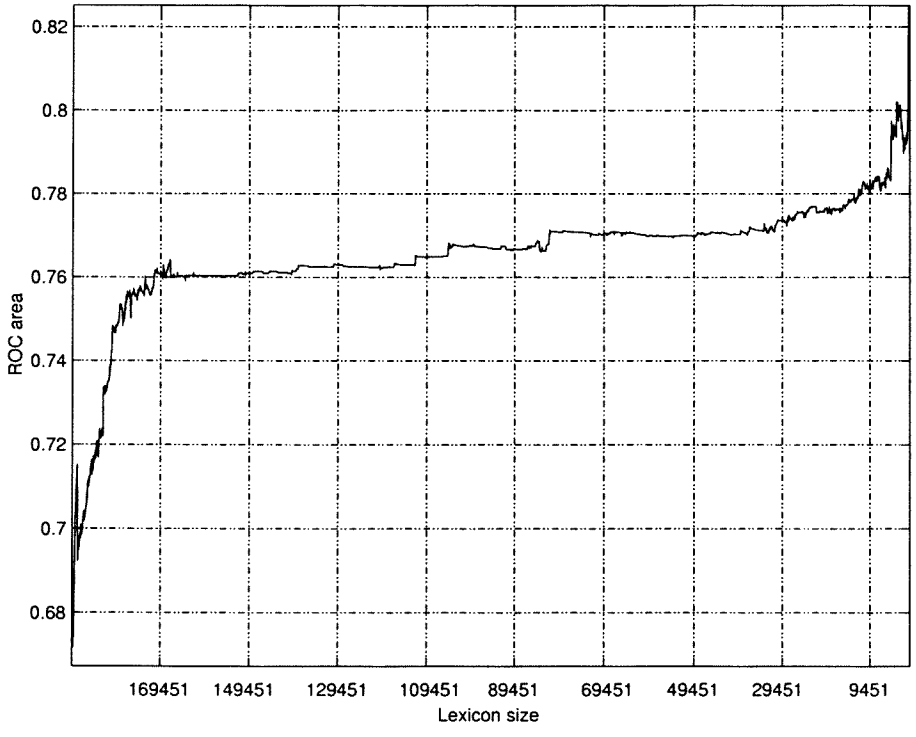


Figure 6.7: *SE's training curve.*

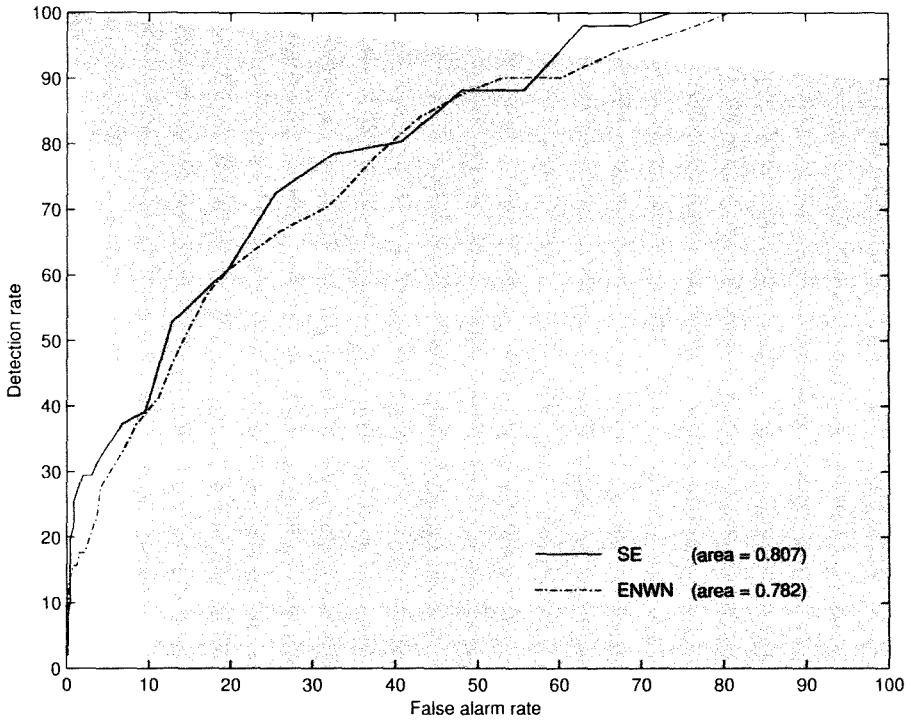


Figure 6.8: *ROC graph: ENWN vs. SE.*

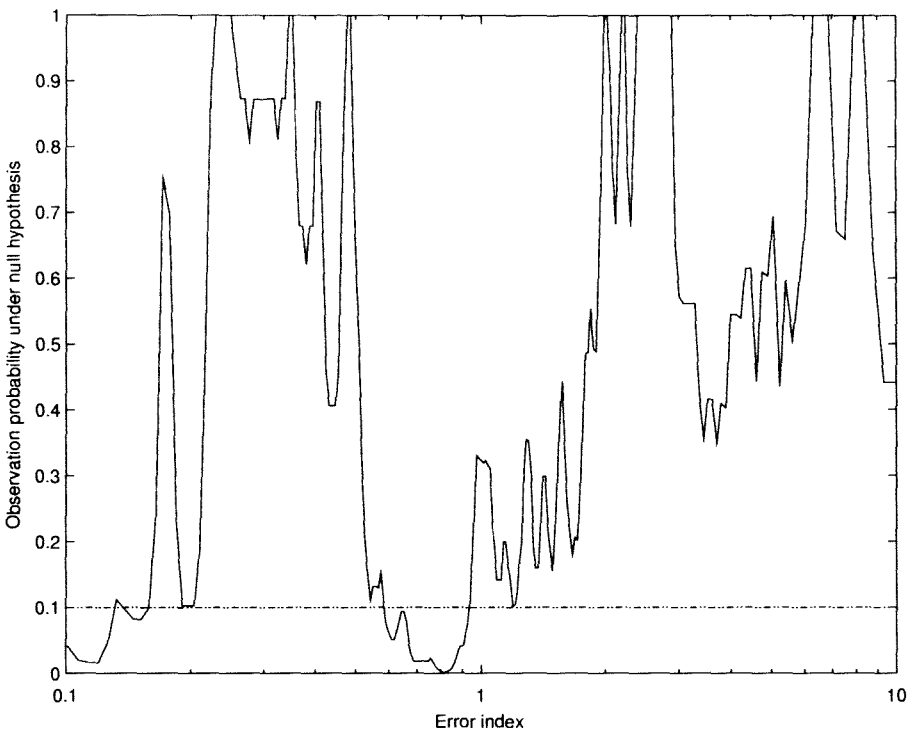


Figure 6.9: *McNemar graph: ENWN vs. SE.*

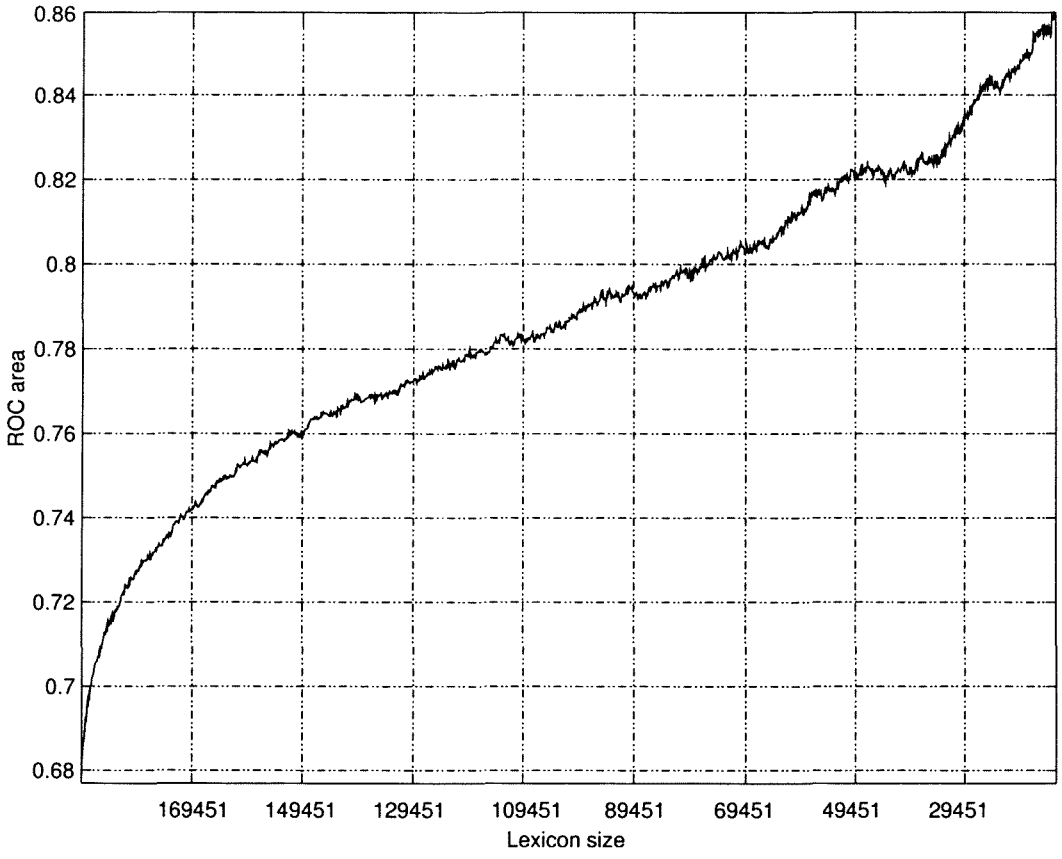


Figure 6.10: *SMART's training curve.*

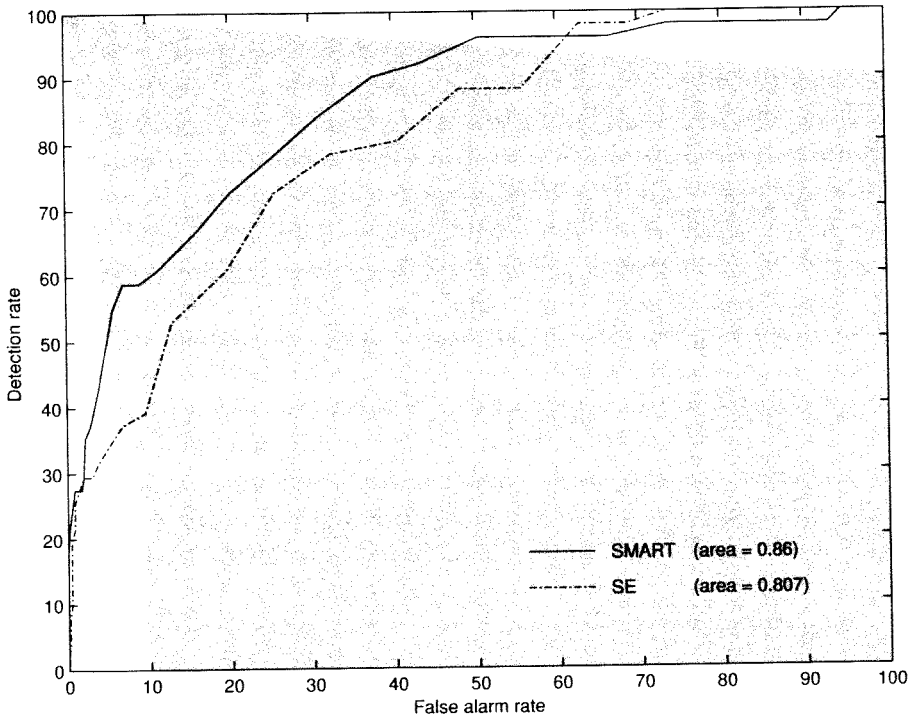


Figure 6.11: *ROC graph: SE vs. SMART.*

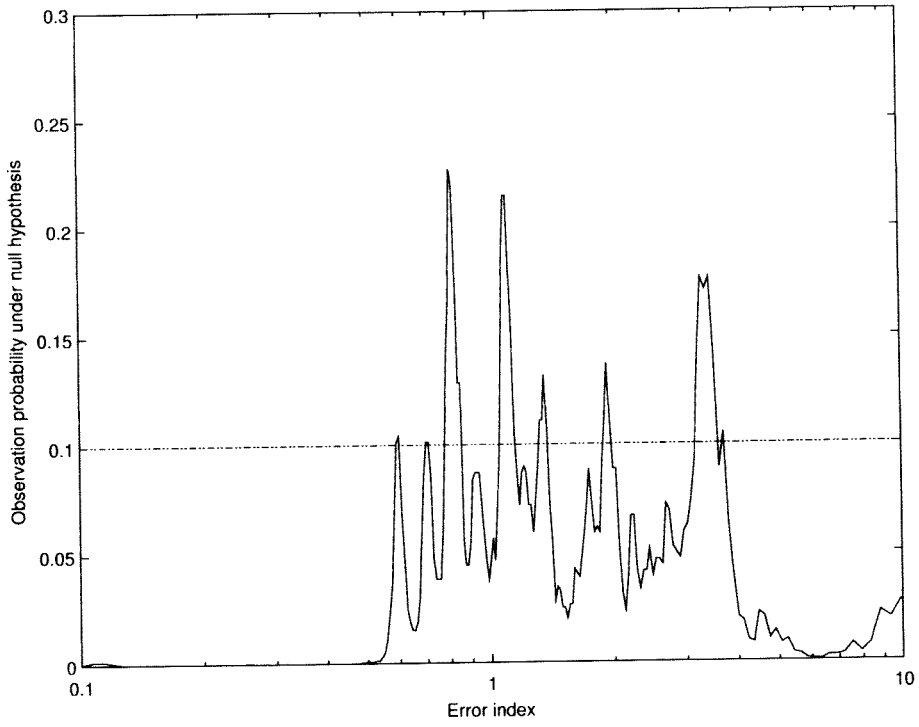


Figure 6.12: *McNemar graph: SE vs. SMART.*

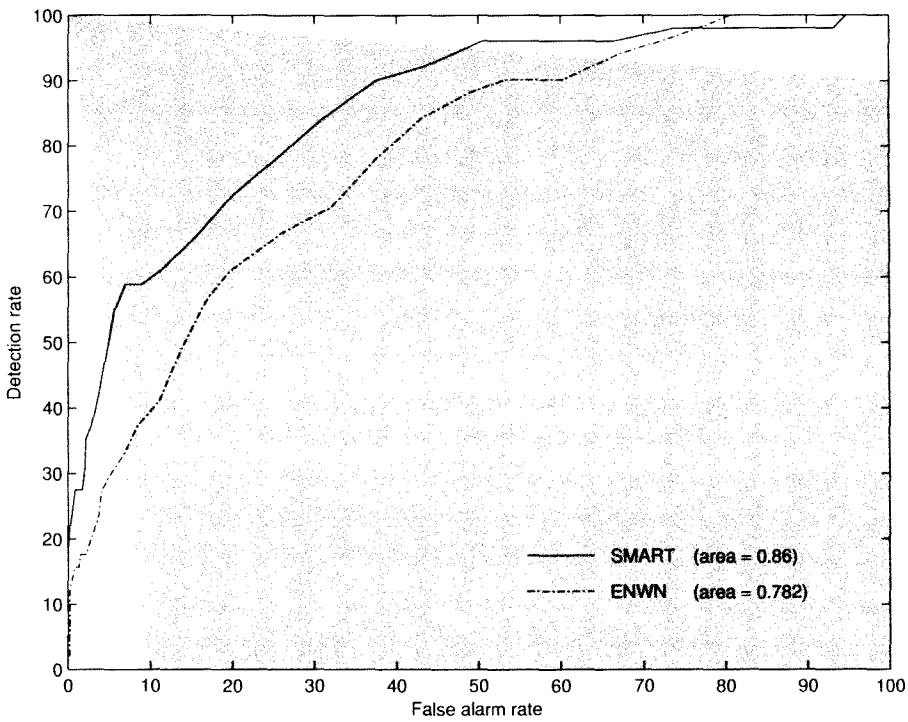


Figure 6.13: *ROC graph: ENWN vs. SMART.*

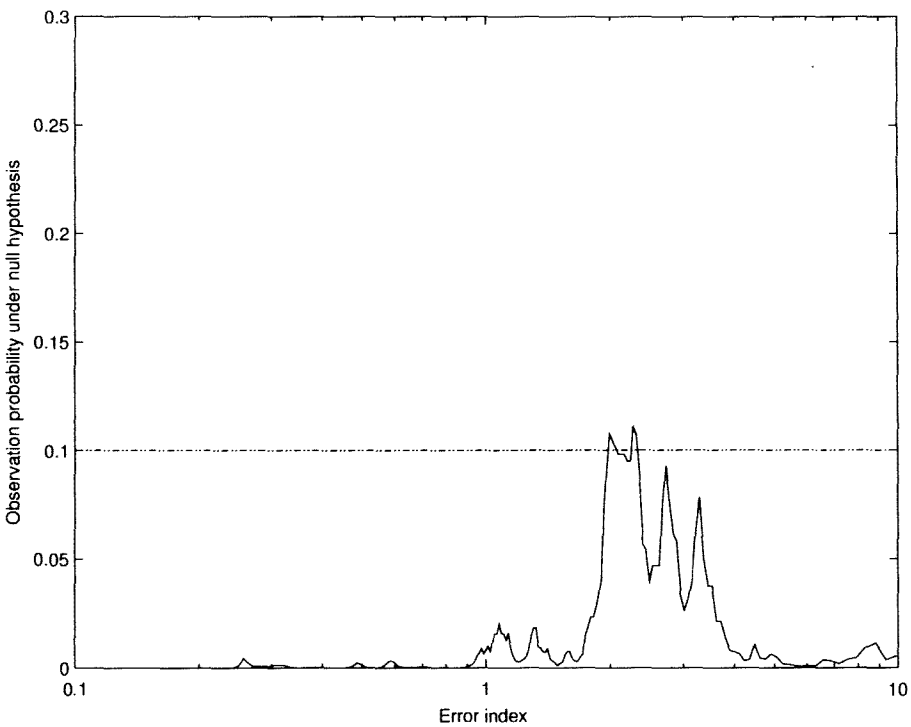


Figure 6.14: *McNemar graph: ENWN vs. SMART.*

Details of the trained lexicons and overall results of ENWN, SE and SMART are presented in Table 6.4. From the table it is clear that the CE distance measure enables SMART to isolate longer lexicon members during pruning of its initial lexicon. Since longer keystrings tend to contain more topic specific information, this may very well account for the significant improvement in topic spotting performance. The fact that the CE distance measure enables SMART to successfully use 4-grams for the purpose of topic discrimination could also explain why SMART's training curve rises steadily over its central region, while ENWN's and SE's remain flat.

Topic Spotter	Feature Extraction	Distance Measure	Size of Trained Lexicon	Lexicon Members of Length 2	3	4	ROC Error Area
ENWN	Hard	Euclidean	1222	141	1048	33	0.218
SE	Soft	Euclidean	405	23	368	14	0.193
SMART	Soft	Cross–entropy	13234	0	1431	11803	0.14

Table 6.4: *Details of the trained lexicons and overall results of ENWN, SE and SMART.*

6.5.1.1 Discussion

It is expected that comparative results between ENWN and SMART could further be improved by using a more sophisticated phoneme recogniser. Previous closed-set tests [24, 25, 26] on the OGI-MLTS Corpus indicated that an improvement in phoneme recognition accuracy not only resulted in improved performance for both algorithms, but also increased the improvement of SMART over ENWN. It thus appears that the advantages obtained by using SMART apply also when phoneme recognition is good.

6.5.2 Extended SMART

SMART was extended by replacing its soft counting strategy with the newly developed one of Section 4.4. The training curve for the extended SMART algorithm is presented in Figure 6.15.

Comparative results between SMART and its extended version are depicted in Figure 6.16. Replacing SMART's soft counting strategy results in a 5% reduction in ROC error area. The outcome of the McNemar tests is shown in Figure 6.17. It is evident from this graph that regions of statistical significance exist between these algorithms.

The improvement of the extended SMART algorithm over ENWN is characterised by a 39% reduction in ROC error area. This is clearly evident from the ROC graph of Figure 6.18. The McNemar graph (Figure 6.19) indicates that the results are statistically significant over *all* error indices that were investigated.

The extended SMART algorithm's performance and details of its trained lexicon are shown in Table 6.5. Note that the number of keystrings used by this algorithm is almost half of that used by SMART (Table 6.4). It consequently produces better results with fewer keystrings.

Topic Spotter	Feature Extraction	Distance Measure	Size of Trained Lexicon	Lexicon Members of Length 2	3	4	ROC Error Area
Extended SMART	Extended Soft	Cross–entropy	7869	0	1049	6820	0.133

Table 6.5: *The extended SMART algorithm's performance and details of its trained lexicon.*

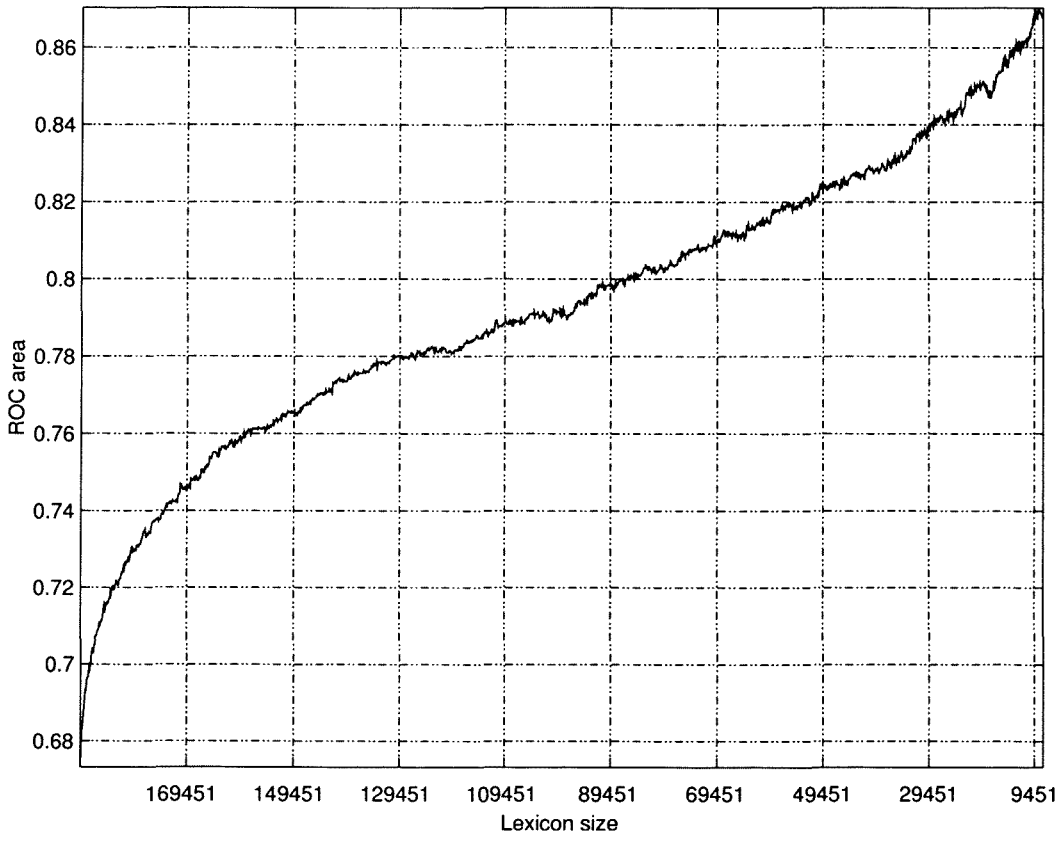


Figure 6.15: *Extended SMART's training curve.*

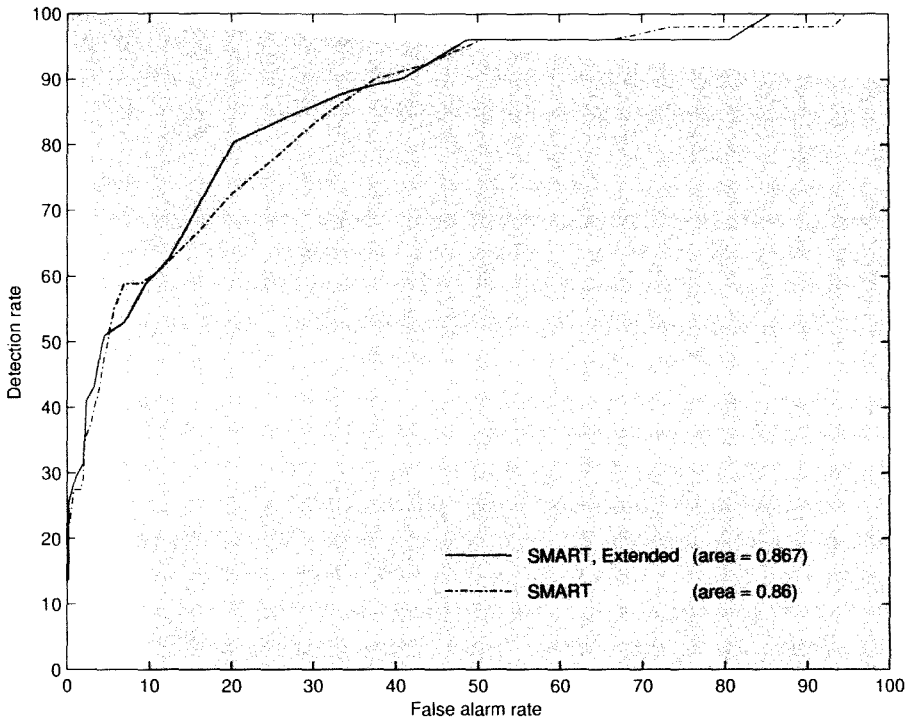


Figure 6.16: *ROC graph: SMART vs. Extended SMART.*

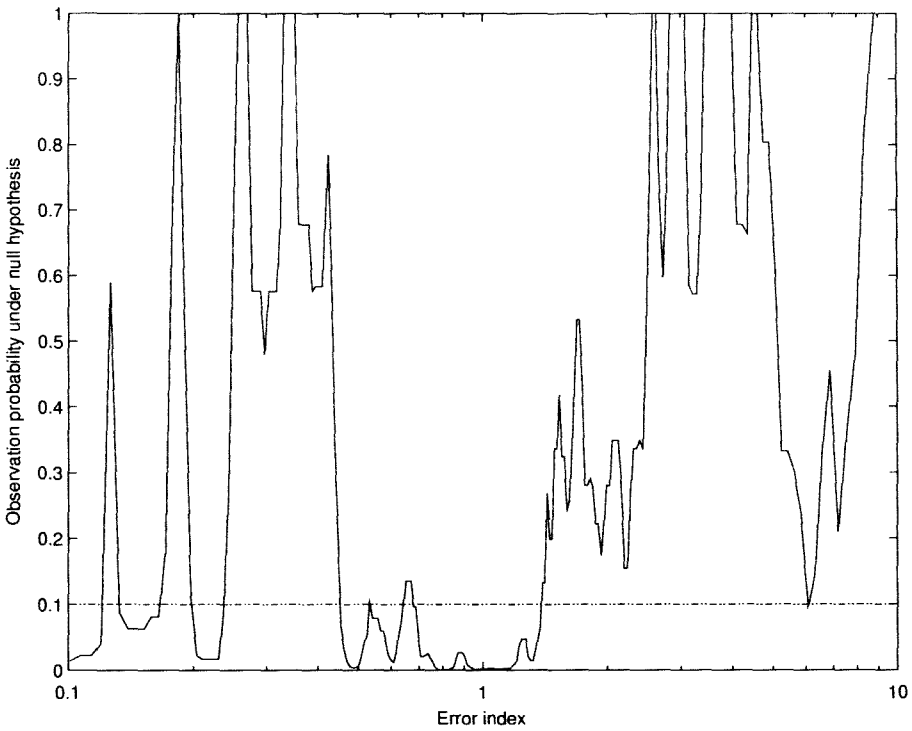


Figure 6.17: *McNemar graph: SMART vs. Extended SMART.*

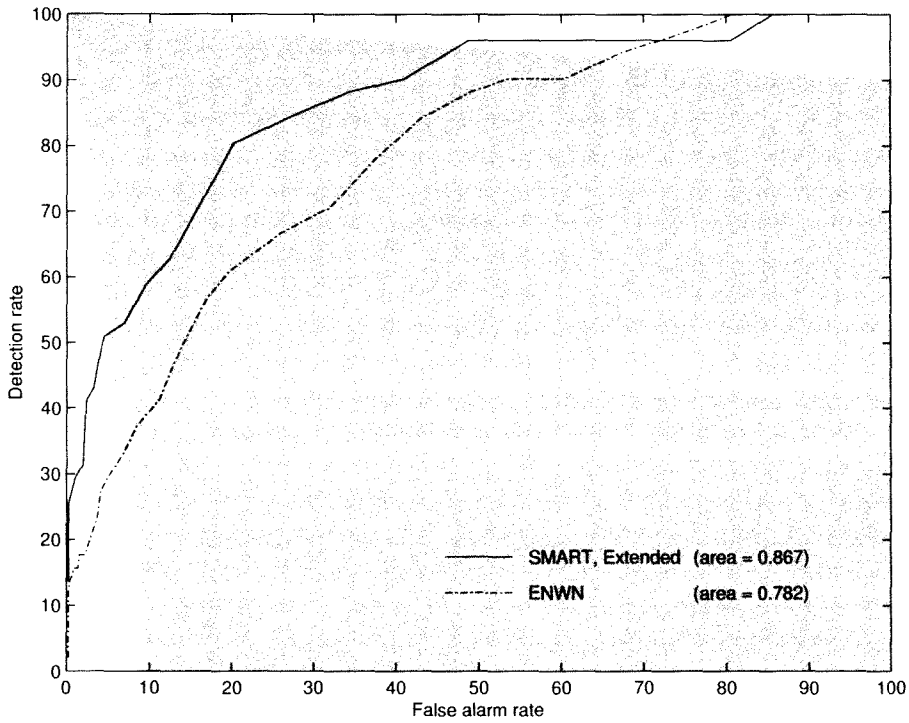


Figure 6.18: *ROC graph: ENWN vs. Extended SMART.*

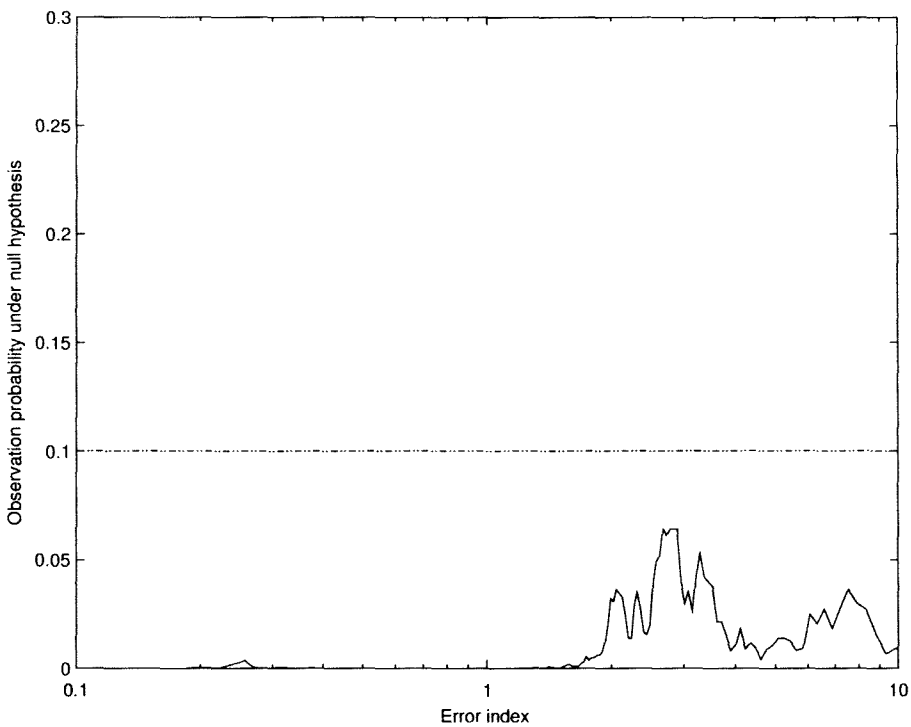


Figure 6.19: *McNemar graph: ENWN vs. Extended SMART.*

6.6 Summary

In this chapter, the experimental setup, hardware specifications, software implementation, and outcome of the most important experiments were presented. The results obtained for the closed-set tests are summarised below:

- The improvement of SMART over ENWN is characterised by a 35.8% reduction in ROC error area. The measured differences are statistically significant over a large range of error indices. The CE distance measure is the main contributor to SMART's success.
- Replacing SMART's soft counting strategy with the newly developed one of Section 4.4 yields a 5% reduction in ROC error area. This is accompanied by regions of statistical significance. The "pigheaded" behaviour of the front end's phoneme-HMMs (Section 4.4.5) probably had a negative impact on the extended SMART algorithm's performance. Much could thus be gained if this problem is addressed.
- Extended SMART also outperforms ENWN. A 39% reduction in ROC error area is observed. The results are statistically significant over *all* error indices that were investigated.

The ROC curves of these topic spotters are shown in Figure 6.20.

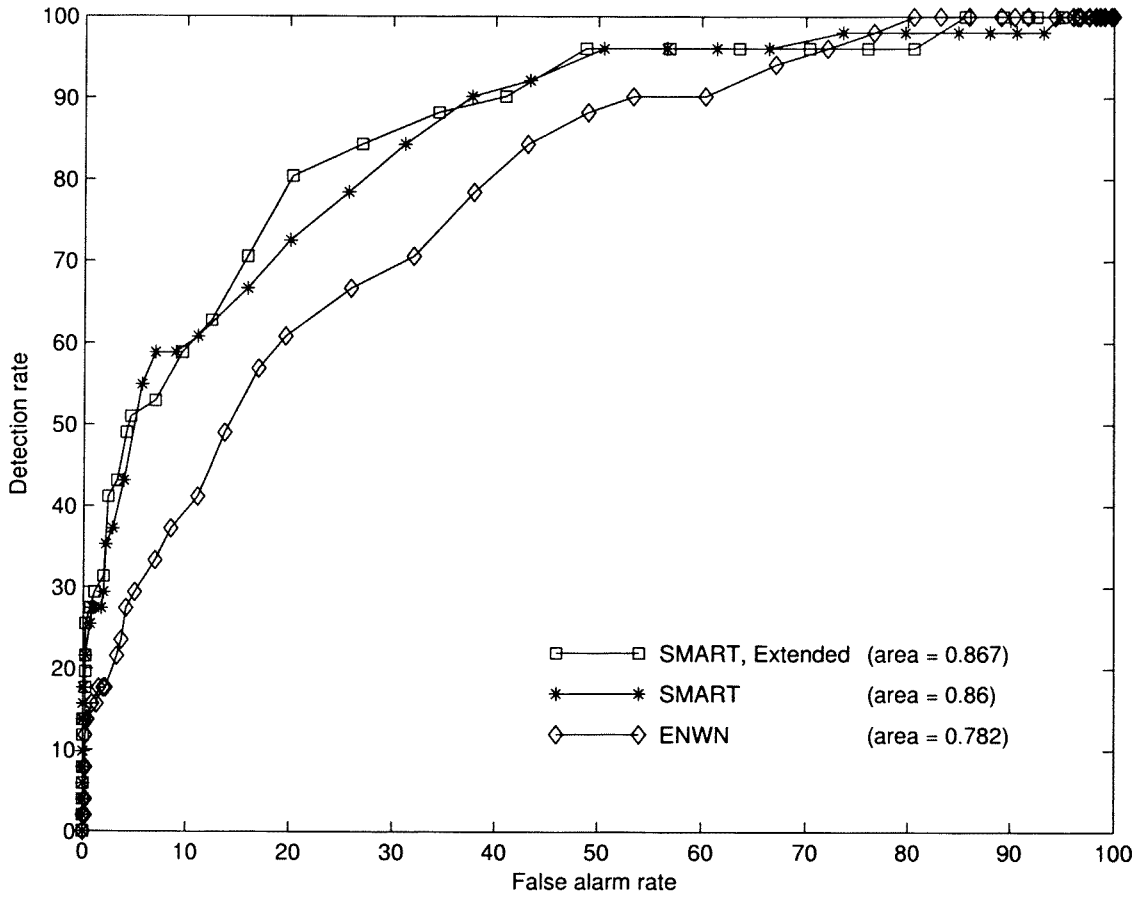


Figure 6.20: *ROC graph: ENWN vs. SMART vs. Extended SMART.*

Chapter 7

Conclusions

The difficulty lies not in the new ideas, but in escaping the old ones, which ramify, for those brought up as most of us have been, into every corner of our minds.

— John M. Keynes

7.1 Summary

During this research, an investigation was conducted into the performance of two competing phoneme-based topic spotting systems, ENWN and SMART. A previous result, obtained on the OGI-MLTS Corpus, suggested that SMART yields a large improvement over ENWN. However, this result was tentative due to the small data set and the use of few topics. The purpose of the research reported here was therefore to determine whether SMART would yield a similar result under a more rigorous test on the larger Switchboard Corpus which is traditionally used in this field. It can be concluded from the experiments that were conducted, that the initial optimism regarding the performance of SMART was justified, since a considerable improvement over ENWN was observed.

An investigation was also conducted into the improvement of SMART. This resulted in

a new soft counting strategy that established a closer connection between the phoneme recognition front end and the rest of the topic spotting system. Low level information, supplied by the phoneme recogniser, is consequently included during the soft counting process in order to obtain a better estimate of the number of times that a lexicon member occurs in the conversation's true phoneme sequence. Although only one of the approaches investigated yielded an improvement, many opportunities still exist for improving SMART. They will be listed in the next section.

7.2 Suggestions for Possible Improvement

Several approaches remain that could possibly improve SMART's performance even further. They are listed below:

- The simultaneous modelling of insertion-, deletion- and substitution-errors could lead to an improvement in performance. To accomplish this, the sequence matchers can be replaced by HMMs. However, this approach is extremely computationally expensive.
- If enough hand-transcribed phonemic transcriptions are available to test the phoneme recognition front end, a context dependent model could be developed of the confusion between the different phonemes. Subsequently, this would enable context dependencies between adjacent phonemes to be taken into account during the topic spotting process.
- If a way could be found to considerably reduce the size of the trained lexicon without decreasing system performance too much, it would allow the use of more sophisticated techniques during the topic spotting process. Consequently, these techniques may compensate for the deterioration in performance that occurs as a result of the smaller lexicon.
- Several topic models remain that could result in an improvement in topic spotting performance. They include both parametric and non-parametric models. However,

provided that enough training data and sufficient computational resources are available, the use of a non-parametric algorithm (e.g. Parzen or K-nearest neighbours [52]) may be the only way to accurately model the data's distribution.

- During this research, short keystrings of no longer than 4 phonemes were used. However, longer keystrings tend to contain more topic-specific information. It consequently follows that much could be gained if a method can be found that isolates longer meaningful keystrings in the data.

- Modelling the simultaneous occurrence of topics in the way described by Imai *et al.* [9] could yield a better result.

Bibliography

[1] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. Upper Saddle River, New Jersey: Prentice-Hall, 1987.

[2] R. C. Rose, E. I. Chang, and R. P. Lippmann, "Techniques for information retrieval from voice messages," *Proc. ICASSP*, pp. 317–320, 1991.

[3] J. R. Rohlicek *et al.*, "Gisting conversational speech," *Proc. ICASSP*, vol. 2, pp. 113–116, 1992.

[4] L. Gillick *et al.*, "Application of large vocabulary continuous speech recognition to topic and speaker identification using telephone speech," *Proc. ICASSP*, vol. 2, pp. 471–474, 1993.

[5] J. McDonough, K. Ng, P. Jeanrenaud, H. Gish, and J. R. Rohlicek, "Approaches to topic identification on the Switchboard Corpus," *Proc. ICASSP*, vol. 1, pp. 385–388, 1994.

[6] J. McDonough and H. Gish, "Issues in topic identification on the Switchboard Corpus," *Proc. ICSLP*, pp. 2163–2166, 1994.

[7] B. A. Carlson, "Unsupervised topic clustering of Switchboard speech messages," *Proc. ICASSP*, pp. 315–319, 1996.

[8] B. Peskin *et al.*, "Improvements in Switchboard recognition and topic identification," *Proc. ICASSP*, pp. 303–306, 1996.

[9] T. Imai, R. Schwartz, F. Kubala, and L. Nguyen, "Improved topic discrimination of broadcast news using a model of multiple simultaneous topics," *Proc. ICASSP*, pp. 727–730, 1997.

[10] J. H. Wright, M. J. Carey, and E. S. Parris, "Improved topic spotting through statistical modelling of keyword dependencies," *Proc. ICASSP*, pp. 313–316, 1995.

[11] J. H. Wright, M. J. Carey, and E. S. Parris, "Topic discrimination using higher-order statistical models of spotted keywords," *Computer Speech and Language*, no. 9, pp. 381–405, 1995.

[12] D. A. James, "A system for unrestricted topic retrieval from radio news broadcasts," *Proc. ICASSP*, pp. 279–282, 1996.

[13] G. J. F. Jones, J. T. Foote, K. Sparck Jones, and S. J. Young, "Robust talker-independent audio document retrieval," *Proc. ICASSP*, pp. 311–314, 1996.

[14] P. Gelin and C. J. Wellekens, "Keyword spotting for video soundtrack indexing," *Proc. ICASSP*, pp. 299–302, 1996.

[15] E. I. Chang, R. P. Lippmann, and D. W. Tong, "Using genetic algorithms to select and create features for pattern classification," *Proc. International Joint Conference on Neural Networks*, vol. 3, pp. 747–752, 1990.

[16] P. N. Garner and A. Hemsworth, "A keyword selection strategy for dialogue move recognition and multi-class topic identification," *Proc. ICASSP*, vol. 3, pp. 1823–1826, 1997.

[17] J. H. Wright, M. J. Carey, and E. S. Parris, "Statistical models for topic identification using phoneme substrings," *Proc. ICASSP*, pp. 307–310, 1996.

[18] S. Greenberg, J. Hollenback, and D. Ellis, "Insights into spoken language gleaned from phonetic transcription of the Switchboard Corpus," *Proc. ICSLP*, pp. s24–s27, 1996.

[19] R. Bellman, *Dynamic Programming.* Princeton, New Jersey: Princeton University Press, 1957.

[20] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–285, 1989.

[21] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition.* Edinburgh University Press, 1990.

[22] J. A. du Preez, *Efficient High-Order Hidden Markov Modelling.* PhD thesis, University of Stellenbosch, Stellenbosch, South Africa, November 1997.

[23] J. A. du Preez, "Efficient training of high-order hidden Markov models using first-order representation," *Computer Speech and Language*, vol. 12, no. 1, pp. 23–39, 1998.

[24] K. Scheffler, "Phoneme-based topic spotting in conversational speech," Master's thesis, University of Stellenbosch, Stellenbosch, South Africa, December 1997.

[25] K. Scheffler and J. A. du Preez, "Stochastic Method for Automatic Recognition of Topics," *Proc. South African Symposium on Communications and Signal Processing*, pp. 67–71, 1998.

[26] K. Scheffler and J. A. du Preez, "Improvements in phoneme-based topic spotting," *Proc. South African Workshop on Pattern Recognition*, pp. 136–139, 1997.

[27] R. Kuhn, P. Nowell, and C. Drouin, "Approaches to phoneme-based topic spotting: An experimental comparison," *Proc. ICASSP*, vol. 3, pp. 1819–1822, 1997.

[28] R. Kuhn and C. Drouin, "Further experiments with phoneme-based topic spotting and first experiments with text classification," tech. rep., Centre de Recherche Informatique de Montréal, October 1996.

[29] E. Nöth, S. Harbeck, H. Niemann, and V. Warnke, "A frame and segment based approach for topic spotting," *Proc. Eurospeech*, pp. 275–278, 1997.

[30] V. Warnke, S. Harbeck, E. Nöth, and H. Niemann, "Topic spotting using subword units," *Tagungsband des 9. Aachener Kolloquiums 'Signaltheorie'*, 1997.

[31] P. Nowell and R. K. Moore, "The application of dynamic programming techniques to non-word based topic spotting," *Proc. Eurospeech*, vol. 2, pp. 1355–1358, 1995.

[32] Q. McNemar, *Psychological Statistics.* New York: John Wiley & Sons, 4th ed., 1969.

[33] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," *Proc. ICASSP*, pp. 532–535, 1989.

[34] Y. K. Muthusamy, R. A. Cole, and B. T. Oshika, "The OGI Multi-Language Telephone Speech Corpus," *Proc. ICSLP*, pp. 895–898, 1992.

[35] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," *Proc. ICASSP*, pp. 517–520, 1992.

[36] M. W. Theunissen, K. Scheffler, and J. A. du Preez, "Phoneme-based topic spotting on the Switchboard Corpus," *Proc. Eurospeech*, pp. 283–287, September 2001.

[37] D. Sankoff and J. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Reading, MA, USA: Addison-Wesley, 1983.

[38] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1998.

[39] P. Z. Peebles, *Probability, Random Variables, and Random Signal Principles.* McGraw-Hill, 3rd ed., 1993.

[40] S. Young, "Large vocabulary continuous speech recognition: A review," *IEEE Signal Processing Magazine*, September 1996.

[41] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications.* Upper Saddle River, New Jersey: Prentice-Hall, 3rd ed., 1996.

[42] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," in *Symposium on Time Series Analysis*, pp. 209–243, John Wiley & Sons, 1963.

[43] S. Furui, "Speaker-independent isolated word recognition using dynamic features of the speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[44] R. Schwarz *et al.*, "Comparative experiments on large vocabulary speech recognition," *Proc. ARPA Workshop on Human Language Technology*, 1993.

[45] *The ICSI Switchboard Phonetic Transcriptions*, November 1996. Documentation on CD-ROM.

[46] L. F. Lamel, R. H. Kassel, and S. Seneff, "Speech database development: Design and analysis of the acoustic-phonetic corpus," *Proc. DARPA Speech Recognition Workshop*, pp. 100–109, February 1986.

[47] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 1641–1648, November 1989.

[48] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 2, pp. 260–269, April 1967.

[49] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, November 1996.

[50] R. van der Merwe, "Variations on statistical phoneme recognition — a hybrid approach," Master's thesis, University of Stellenbosch, Stellenbosch, South Africa, December 1997.

[51] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons, 1973.

[52] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* London, Endland: Prentice-Hall, 1982.

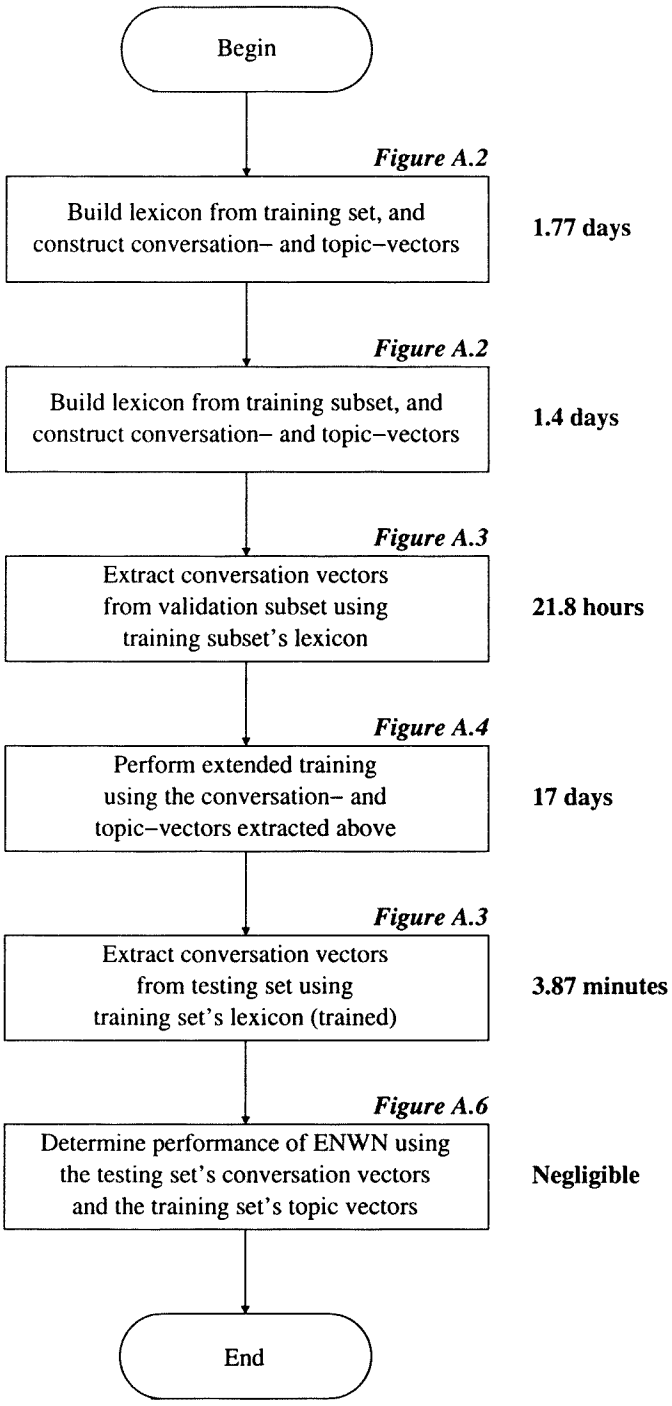
Appendix A

Flow diagrams of the Optimisation of ENWN and SMART

Flow diagrams of ENWN and SMART, before and after they were optimised, will be shown in this appendix. Scheffler's implementation of ENWN and SMART is illustrated in Figures A.1–A.9, while flow diagrams of their new implementation are depicted in Figures A.10–A.17. Note that:

- The overall flow diagram of Scheffler's implementation of ENWN is given in Figure A.1.
- The overall flow diagram of Scheffler's implementation of SMART is given in Figure A.7.
- The overall flow diagram of the new implementation of ENWN is given in Figure A.10.
- The overall flow diagram of the new implementation of SMART is given in Figure A.16.

Simulation times are also shown in these flow diagrams.



Total simulation time = 21.1 days

Figure A.1: *Overall flow diagram of ENWN. Estimated simulation times are also shown — phoneme n-grams in the length range 2 to 4 were allowed, and the recurrence threshold was set to 2. (Scheffler's implementation)*

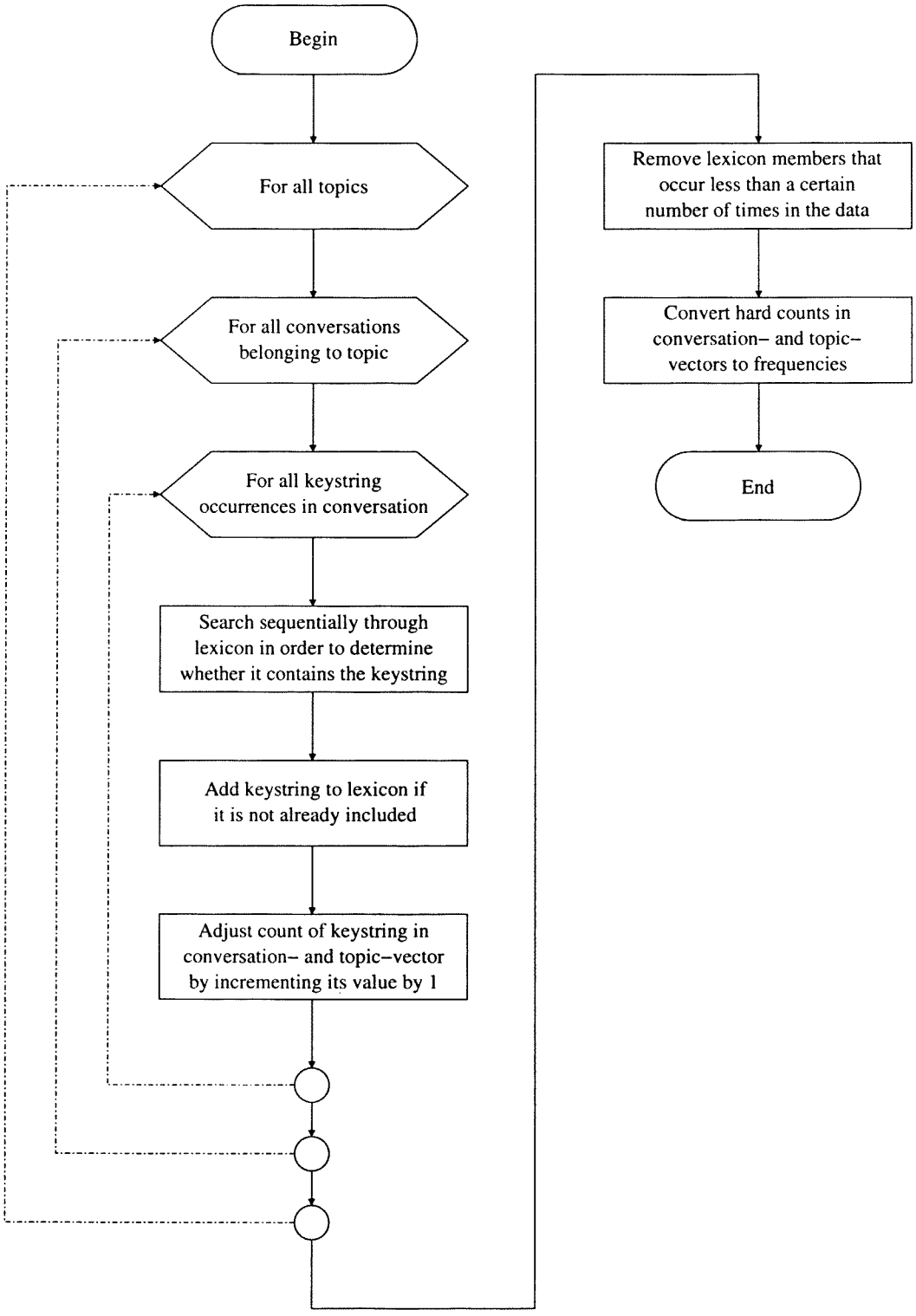


Figure A.2: *Lexicon initialisation. Conversation- and topic-vectors are also generated. Hard counts are employed. (Scheffler's implementation)*

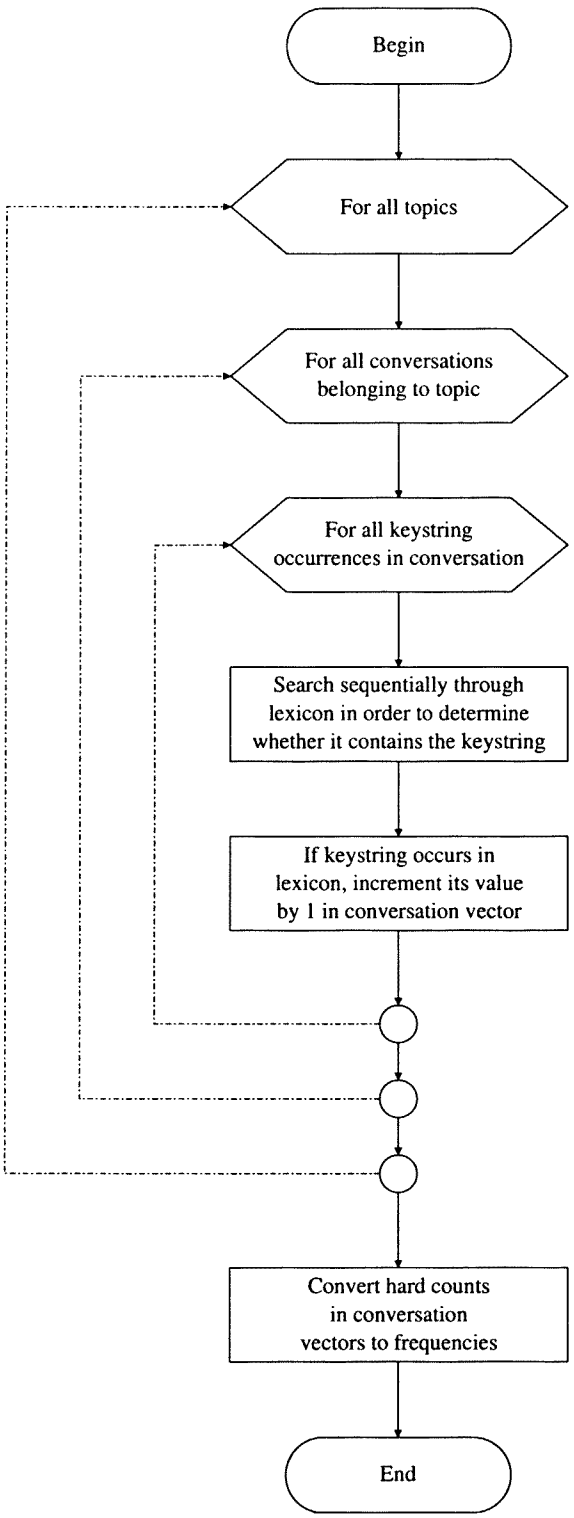


Figure A.3: *Extracting conversation vectors for a given lexicon. Hard counts are employed. (Scheffler's implementation)*

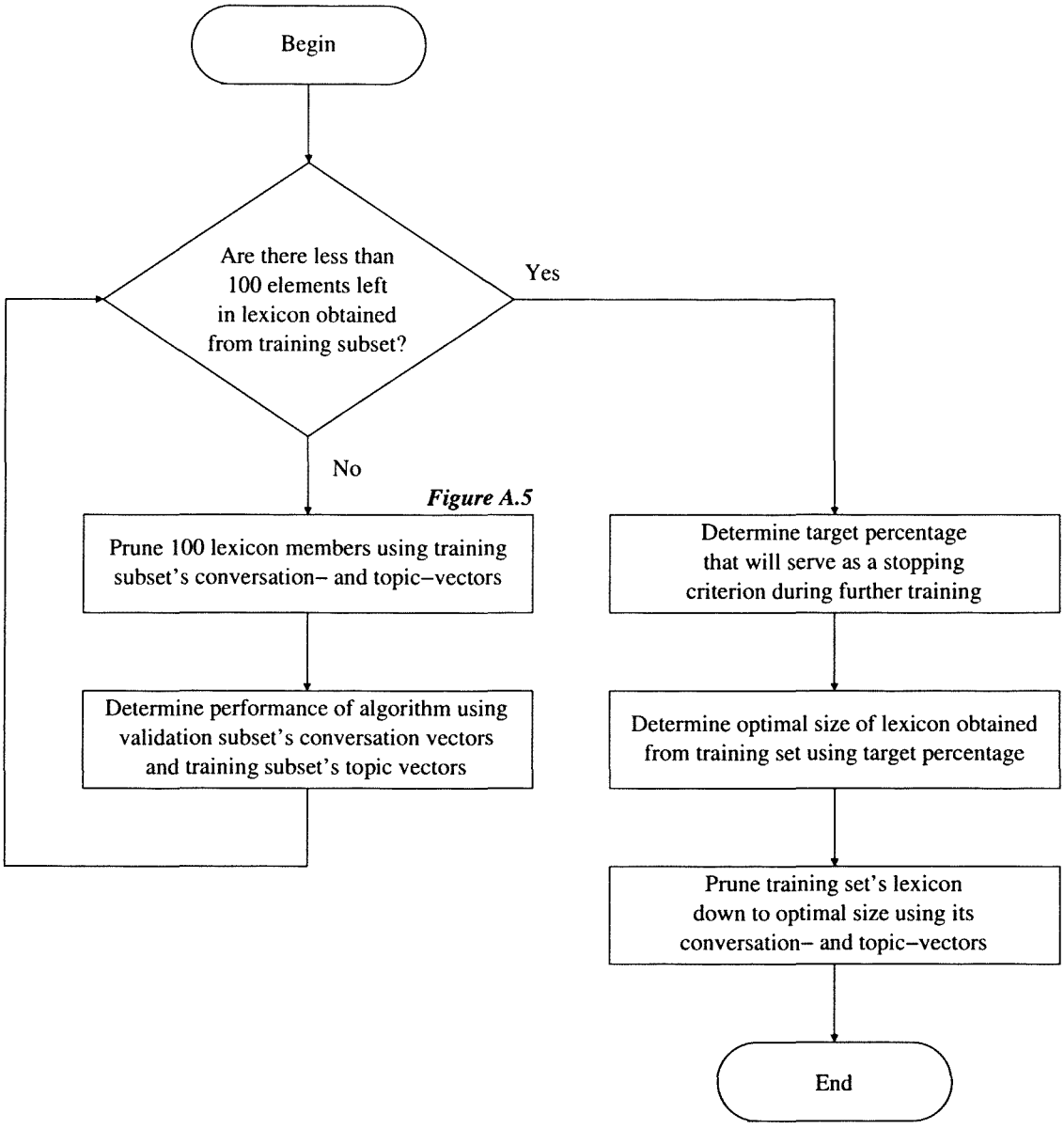


Figure A.4: *Extended training. (Scheffler's implementation)*

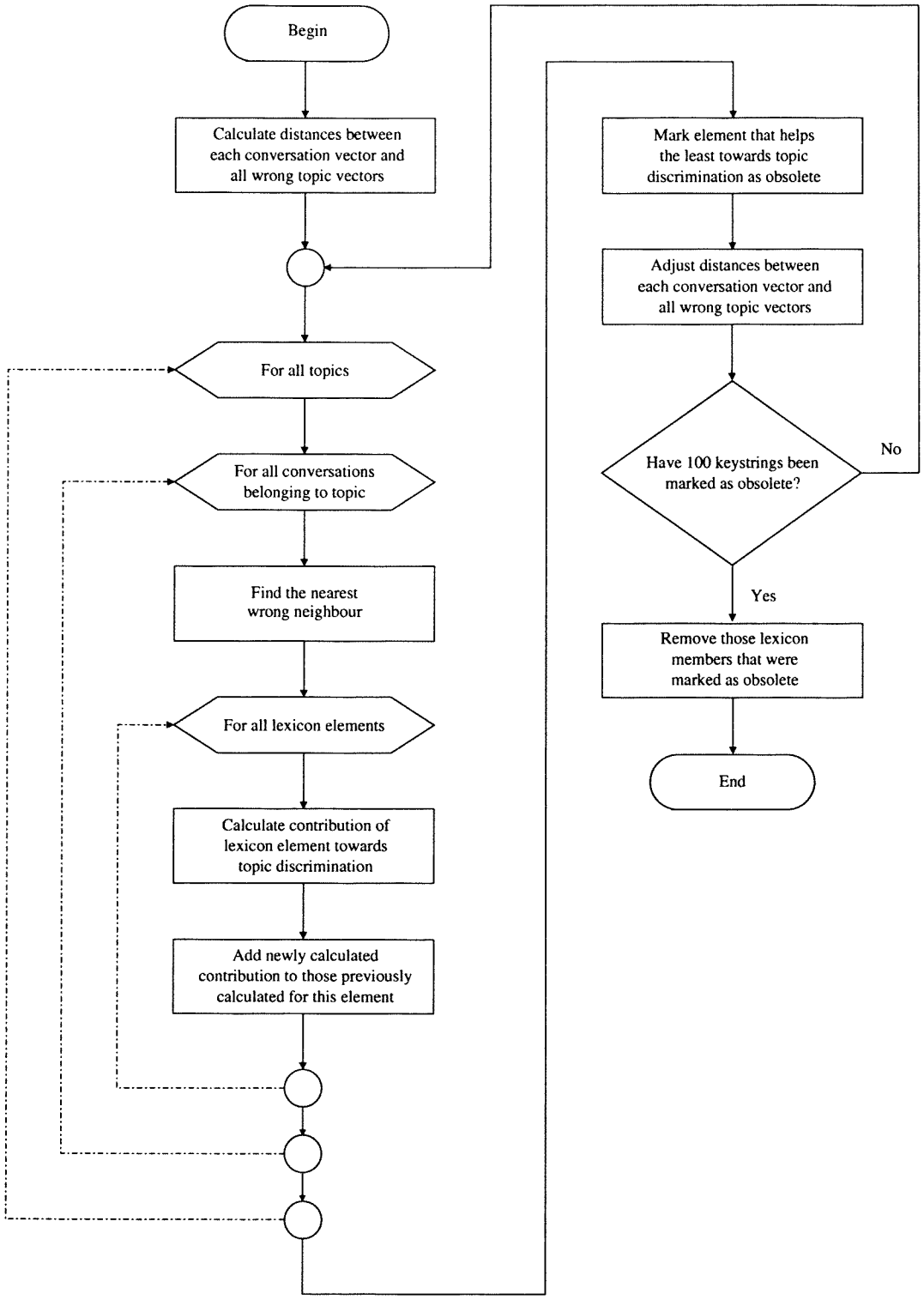


Figure A.5: *Pruning lexicon members. (Scheffler's implementation)*

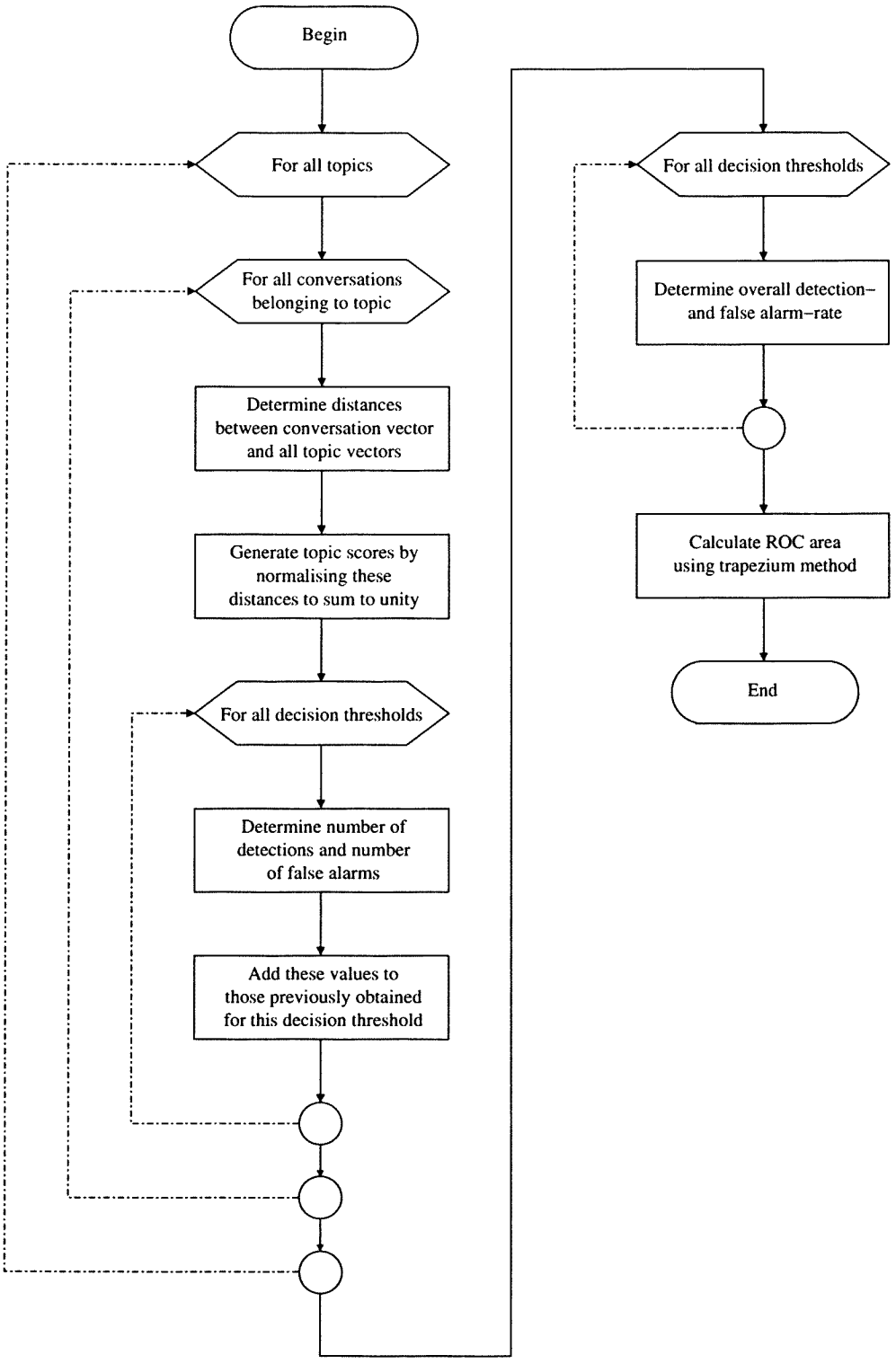


Figure A.6: *Determining an algorithm's performance. (Scheffler's implementation)*

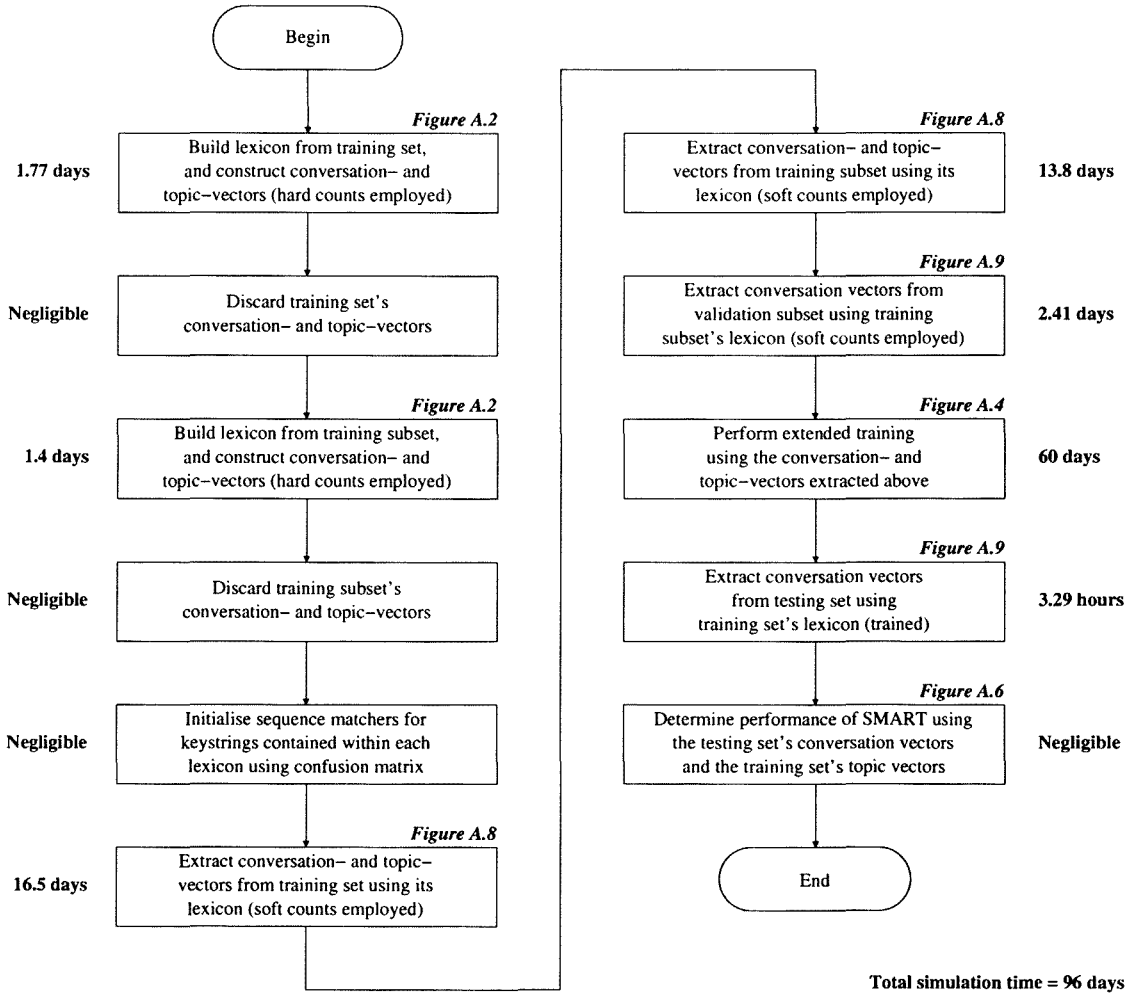


Figure A.7: *Overall flow diagram of SMART. Estimated simulation times are also shown — phoneme n-grams in the length range 2 to 4 were allowed, and the recurrence threshold was set to 2. (Scheffler's implementation)*

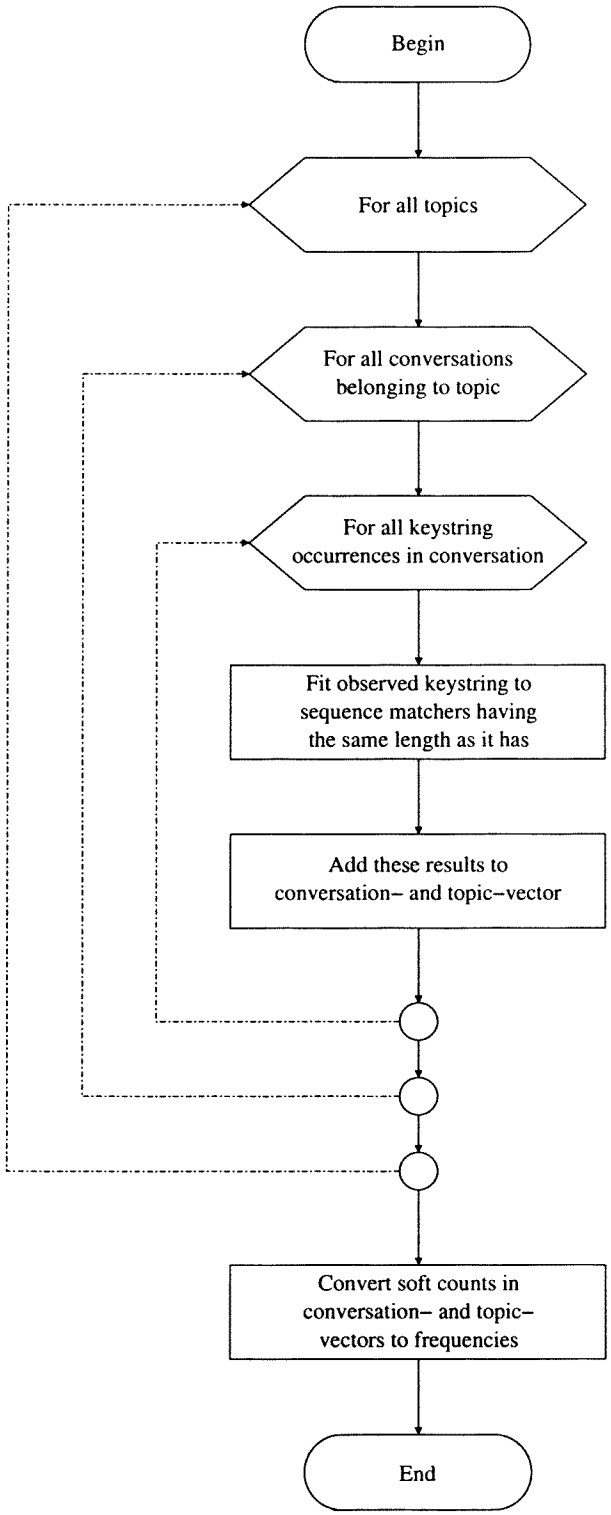


Figure A.8: *Extracting conversation- and topic-vectors for a given lexicon. Soft counts are employed. (Scheffler's implementation)*

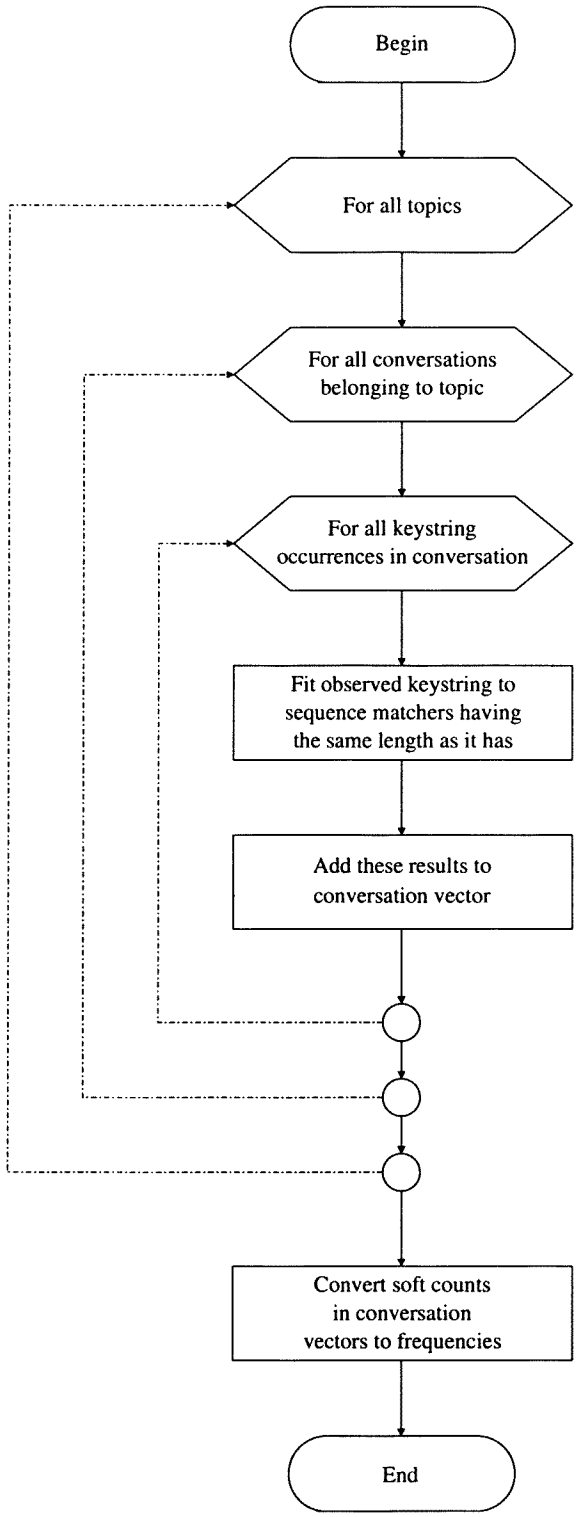


Figure A.9: *Extracting conversation vectors for a given lexicon. Soft counts are employed. (Scheffler's implementation)*

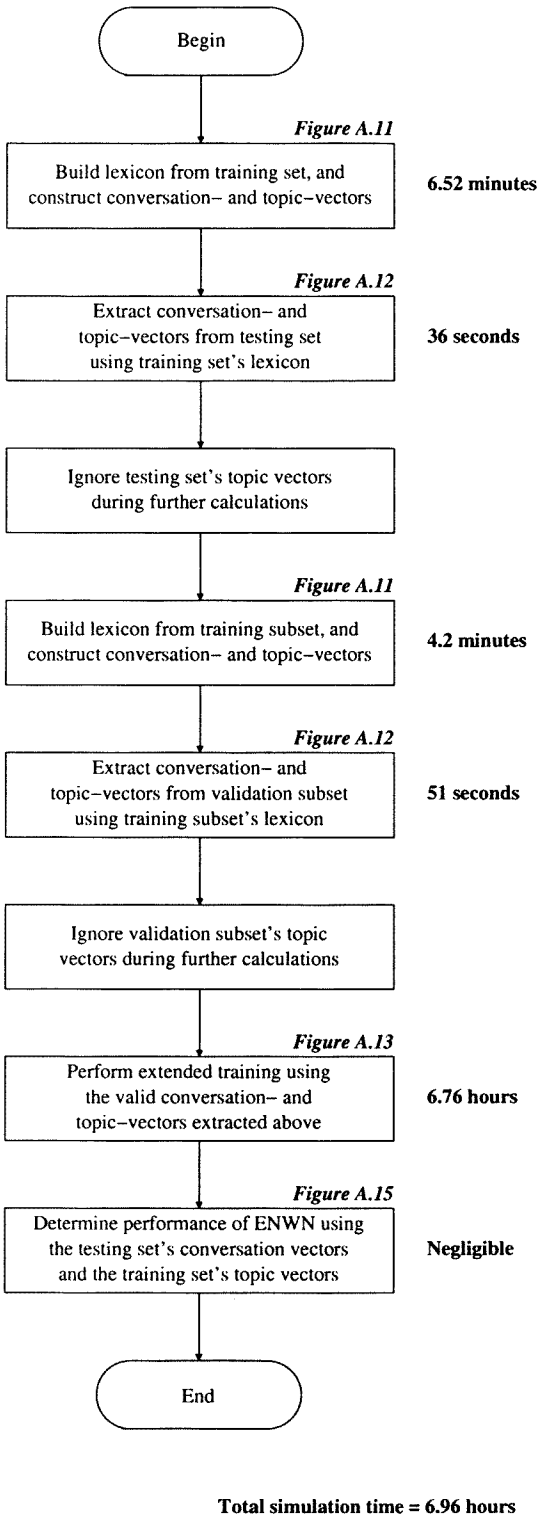


Figure A.10: *Overall flow diagram of ENWN. Simulation times are also shown — phoneme n-grams in the length range 2 to 4 were allowed, and the recurrence threshold was set to 2. (new implementation)*

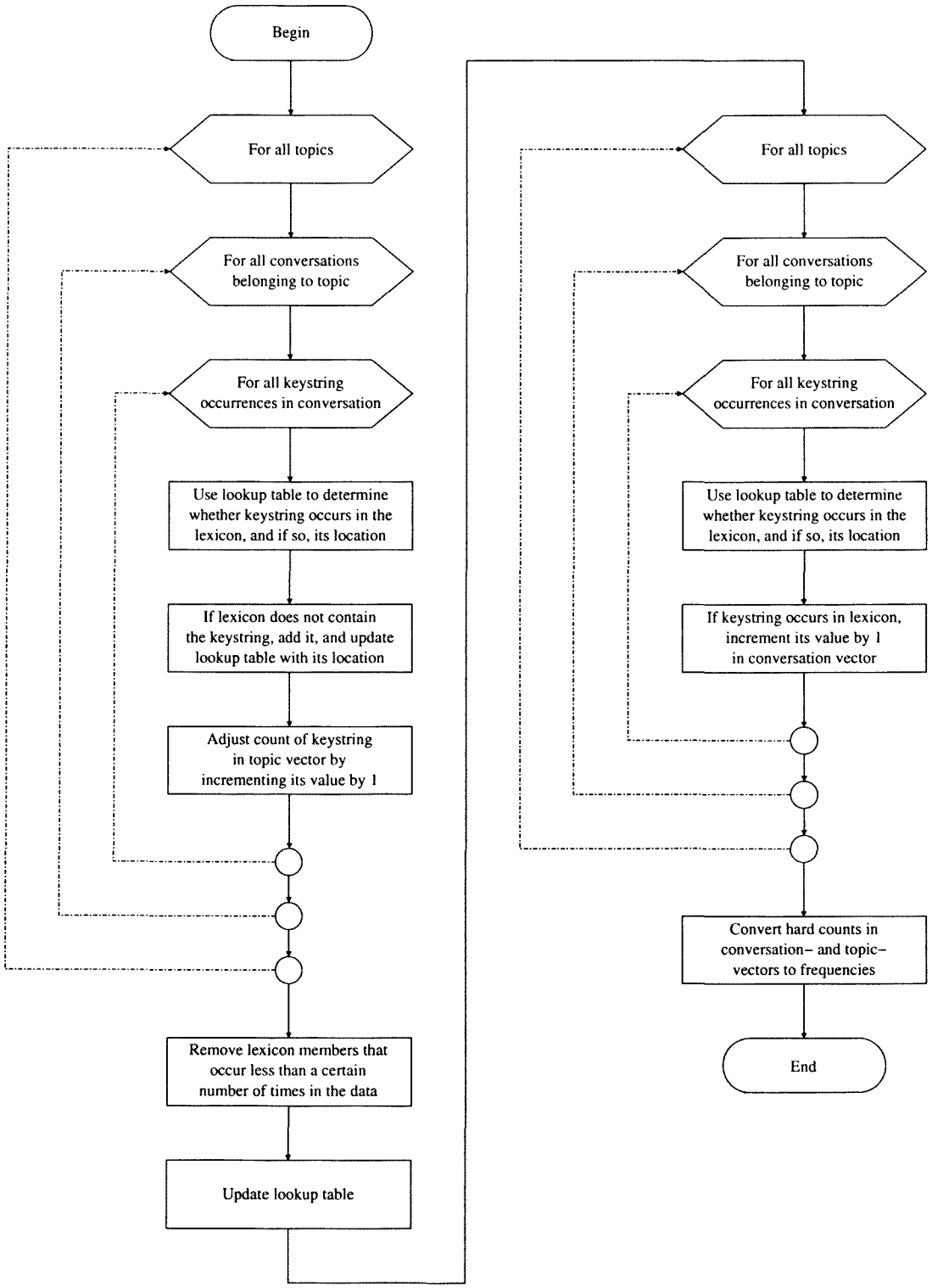


Figure A.11: *Lexicon initialisation. Conversation- and topic-vectors are also generated. Hard counts are employed. (new implementation)*

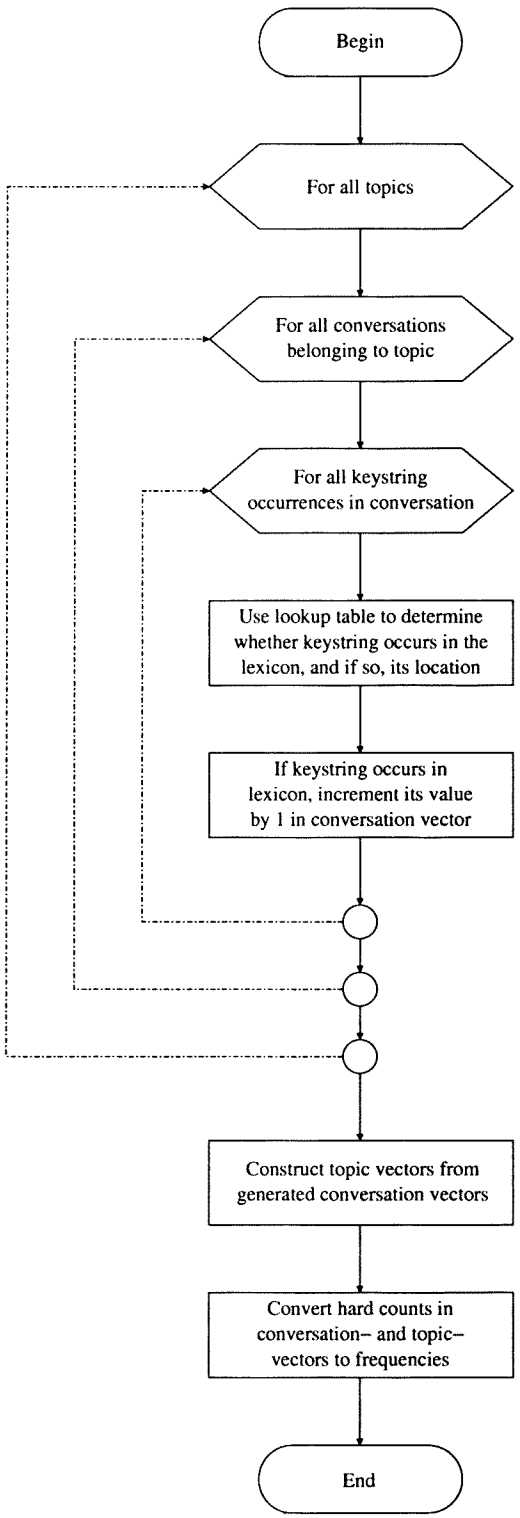


Figure A.12: *Extracting conversation- and topic-vectors for a given lexicon. Hard counts are employed. (new implementation)*

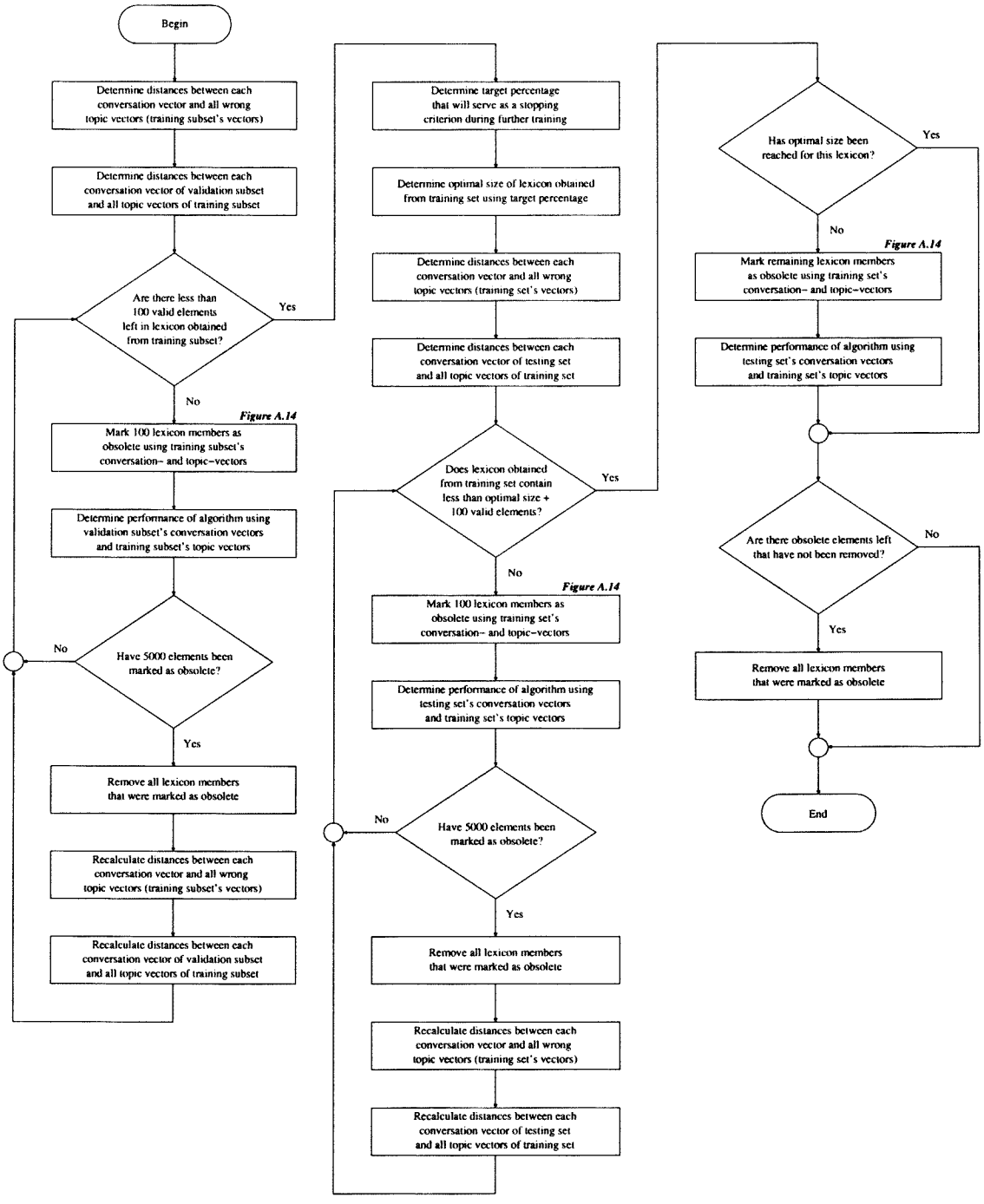


Figure A.13: *Extended training. (new implementation)*

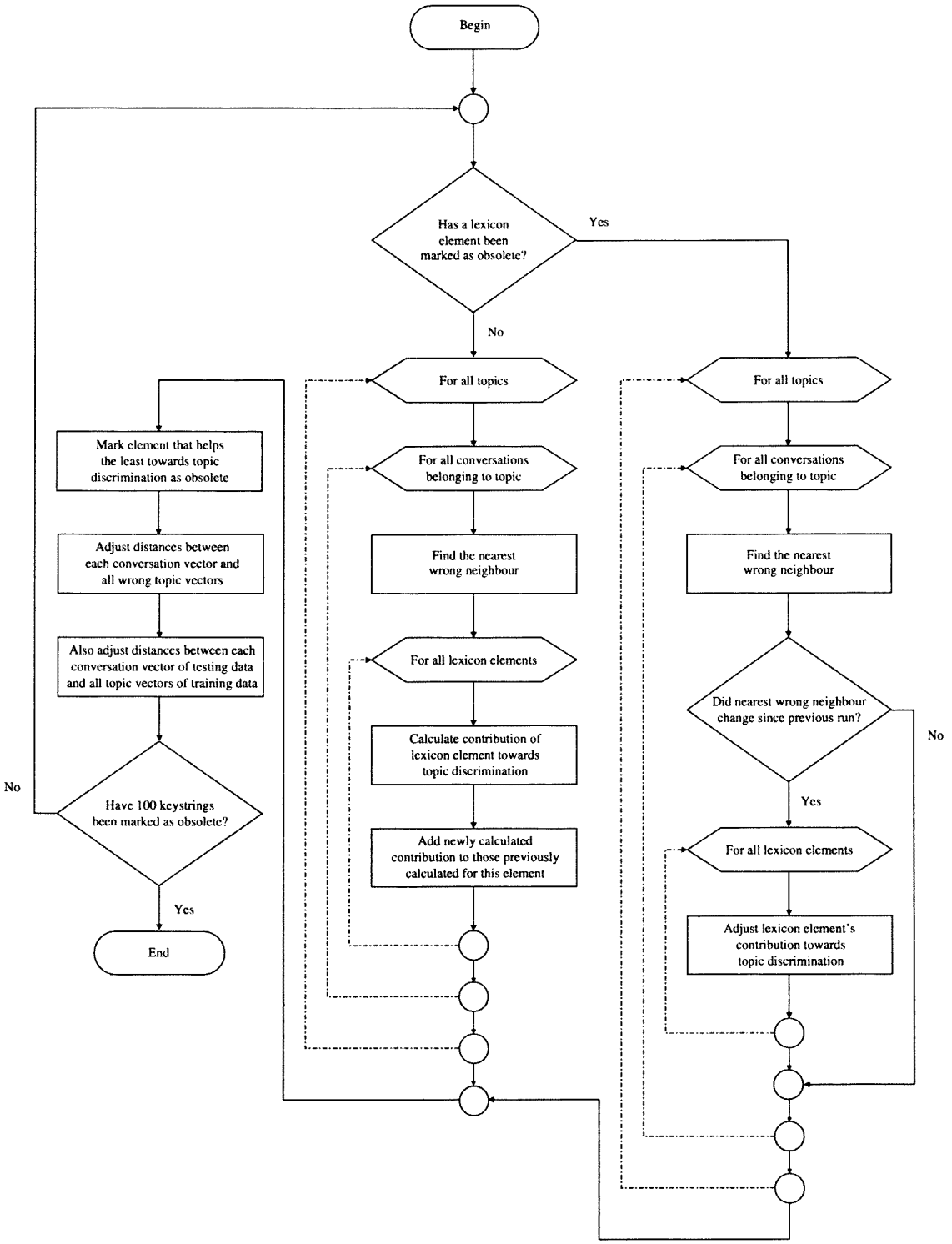


Figure A.14: *Marking lexicon members as obsolete. (new implementation)*

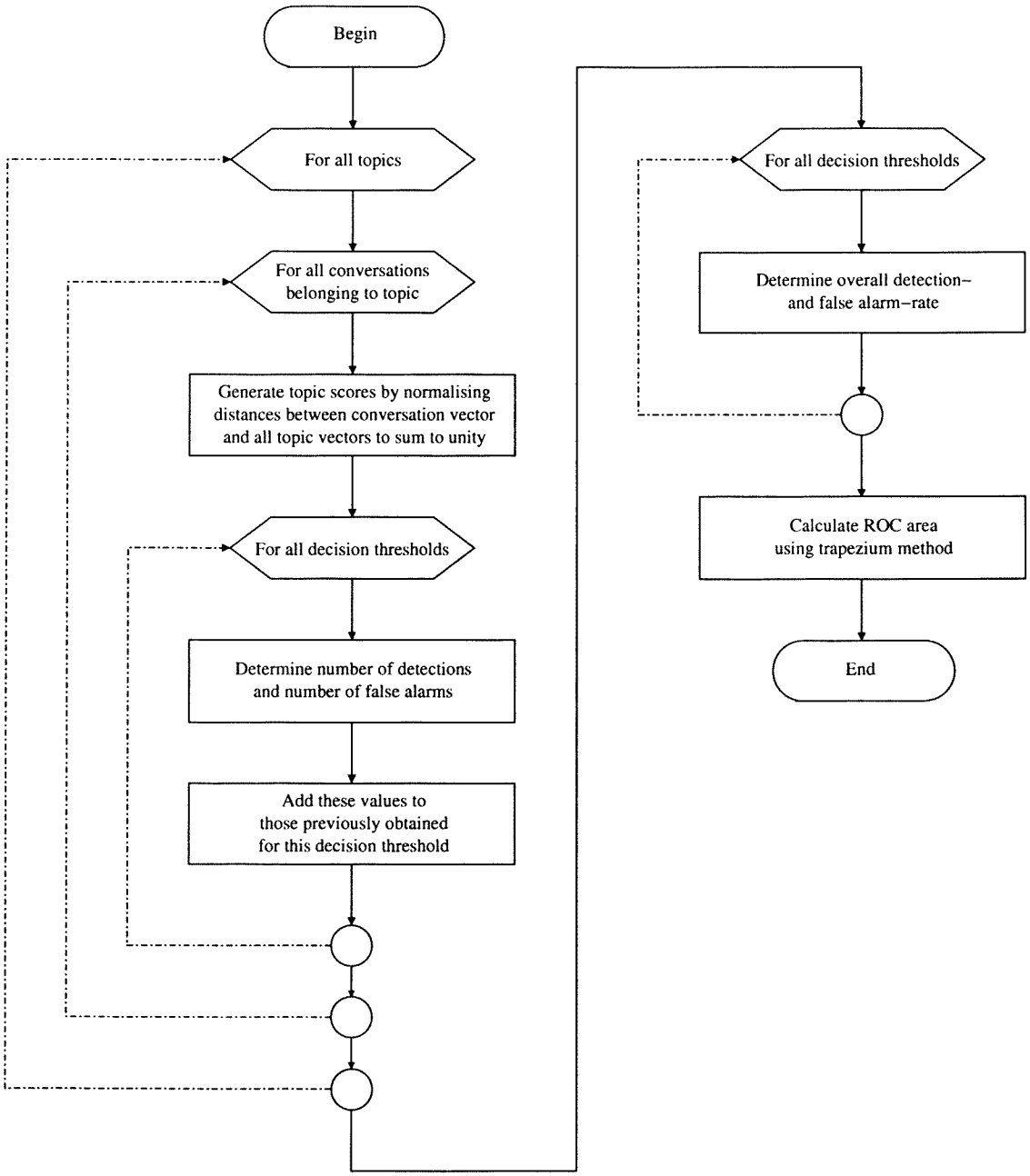


Figure A.15: *Determining an algorithm's performance. (new implementation)*

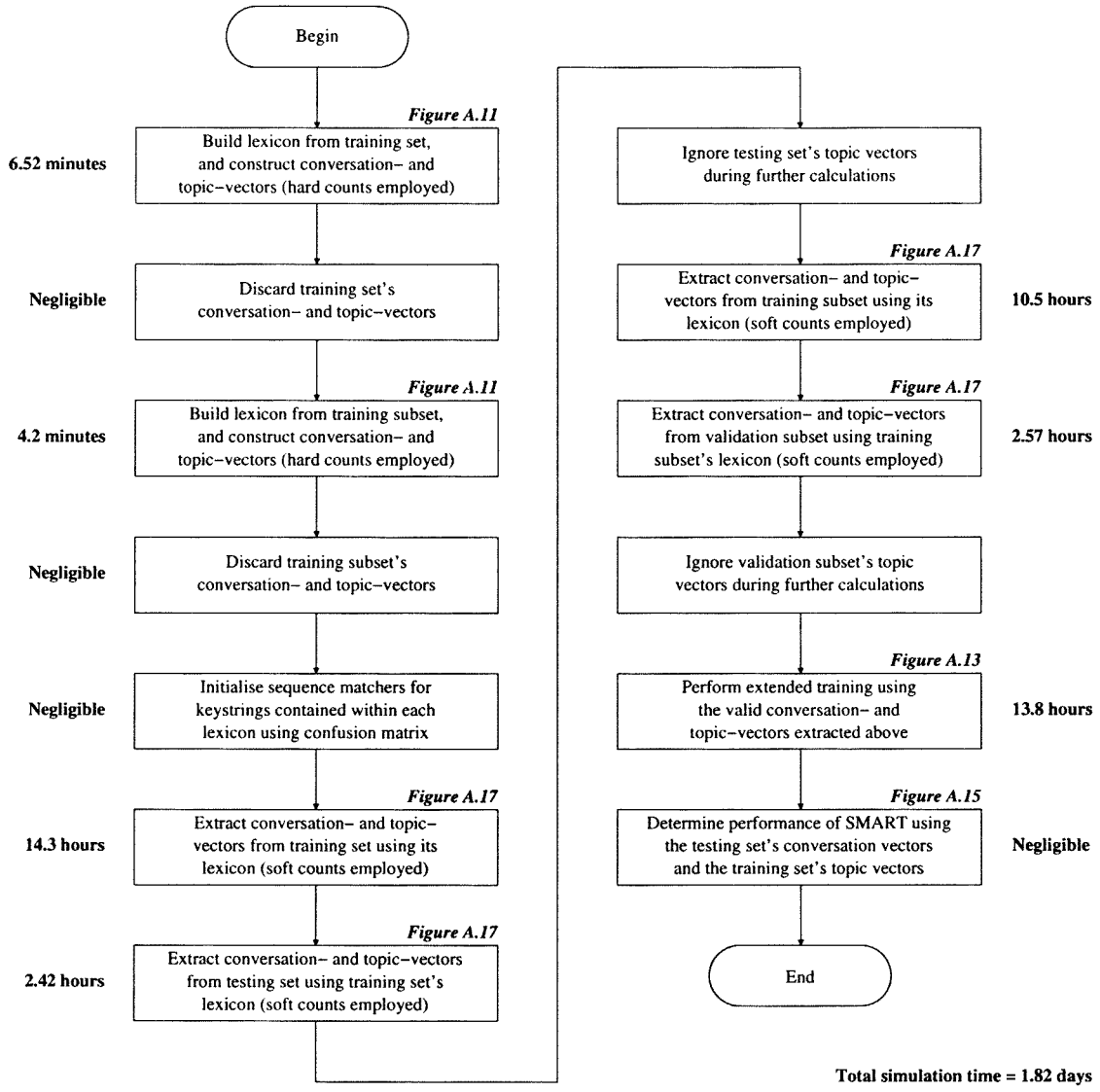


Figure A.16: *Overall flow diagram of SMART. Simulation times are also shown — phoneme n-grams in the length range 2 to 4 were allowed, and the recurrence threshold was set to 2. (new implementation)*

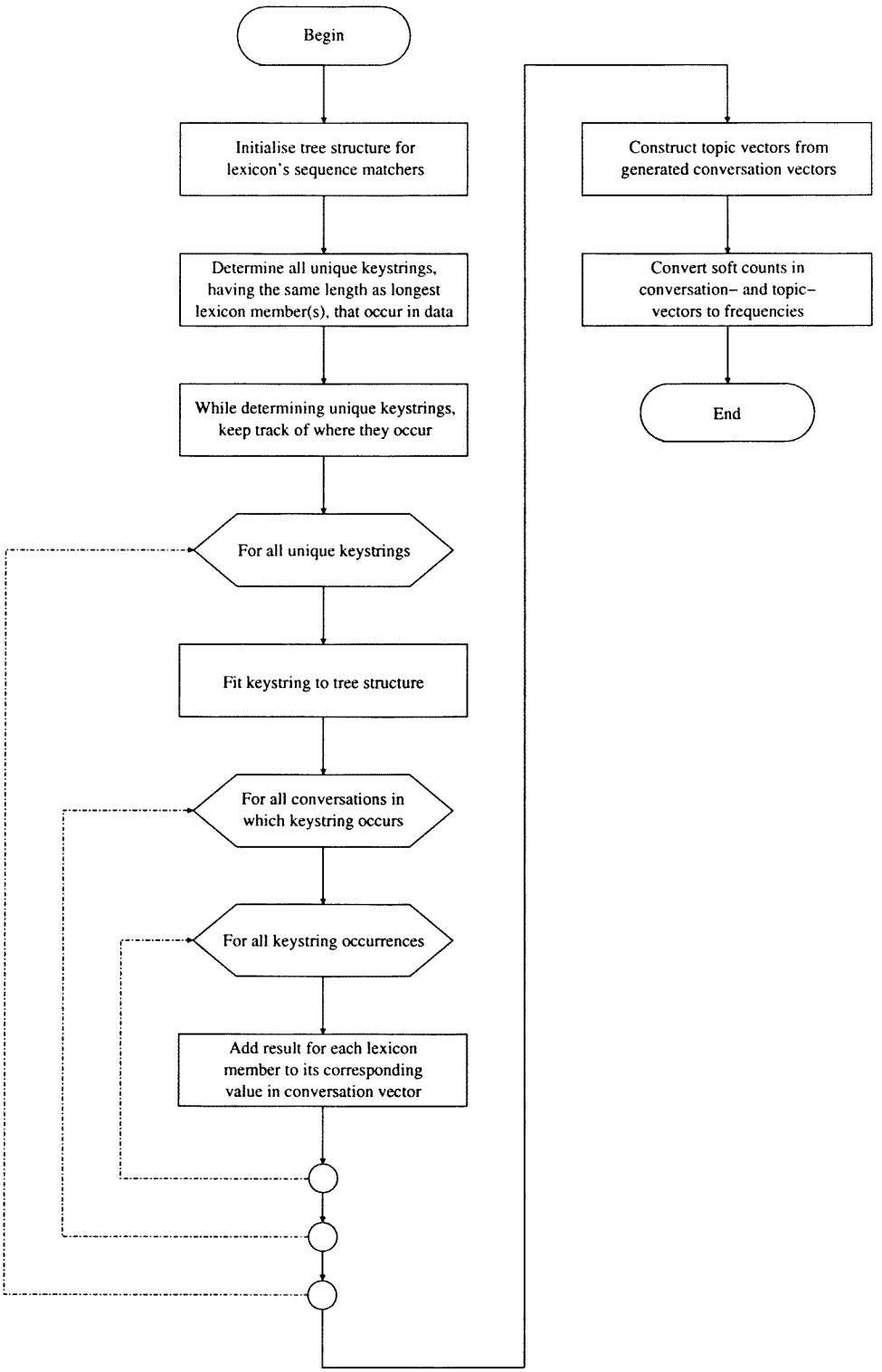


Figure A.17: *Extracting conversation- and topic-vectors for a given lexicon. Soft counts are employed. (new implementation)*