Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Automatic Media Monitoring Using Stochastic Pattern Recognition Techniques

Uri Iurgel

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. sc.techn.(ETH) Andreas Herkersdorf
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll
	2. Univ.-Prof. Dr.-Ing. Wolfgang Utschick

Die Dissertation wurde am 20.06.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 27.04.2006 angenommen.

Acknowledgements

This thesis is a result of my work as a research assistant at the Institute for Information Technology at the Gerhard-Mercator-University of Duisburg (now University of Duisburg-Essen).

I would like to start by thanking my advisor Prof. Gerhard Rigoll. He has supervised and supported my thesis even after his change from University of Duisburg to Munich Unversity of Technology.

I would also like to express my gratitude to Prof. Wolfgang Utschick for agreeing to join my thesis committee.

I have much profited from the cooperation and support by Prof. Hans-Dieter Kochs and his team (especially Daniela Lücke-Janssen) at the University of Duisburg-Essen.

I would also like to thank all my colleagues for many fruitful discussions and ideas. A special thanks goes to Dr. Martha Larson, Dr. Andreas Kosmala, and Dr. Stefan Eickeler for proof-reading.

Ido Iurgel, Dr. Michal Or-Guil, and Rahel Hoffmann also helped in finalising this thesis.

Sincere thanks to my wife and my children for their never-ending patience and support.

Kurzfassung

Informationen sind von entscheidender Bedeutung, sowohl für Industrie und Behörden als auch für Privatleute. Automatische Verfahren zur Auswahl und Verbreitung von Informationen ermöglichen es der Medienauswertung, deutlich mehr Medienquellen abzudecken; insbesondere durch eine höhere Kosteneffizienz und durch Service rund um die Uhr.

Die vorliegende Arbeit untersucht, inwiefern die – bislang hauptsächlich manuell vorgenommene – professionelle Medienauswertung automatisiert unterstützt werden kann. Drei Hauptmodule sind hierzu notwendig: Spracherkennung, Themensegmentierung und Themenklassifizierung. Die Forschungsergebnisse bezüglich der einzelnen Module des Demonstrators werden zusammen mit den erreichten Innovationen dargestellt. Die Leistungsfähigkeit sowohl der Module als auch des gesamten Systems wird anhand von ausführlichen Tests untersucht.

Der Schwerpunkt dieser Arbeit liegt auf deutschsprachigen Nachrichtensendungen. Mittels visueller Indizierungsverfahren werden Themengrenzen in Fernsehnachrichten bestimmt. Ein Spracherkenner wandelt die Audiosignale in Texte um, welche von einem Themenklassifizierer auf das Vorkommen von vorgegebenen Themen überprüft werden. Es werden statistische Klassifizierer wie Hidden Markov Modelle und Support Vector Machines (SVMs) verwendet. Ein Beitrag dieser Arbeit liegt in der Vorstellung von neuartigen Couplern zu SVMs, die Vorteile gegenüber bekannten Couplern besitzen.

Ein weiteres behandeltes Thema ist die unüberwachte Themenfindung (Unsupervised Topic Discovery), die in der Literatur fast gänzlich unbeachtet bleibt. Sie erlaubt es, Stichwörter ohne eine vorgegebene Themenliste oder ohne Trainingsbeispiele zu finden.

Abstract

Information is of strategic importance for business and governmental agencies, but also for individual citizens. The use of automatic methods for selection and dissemination of information would enable media monitoring companies to cover a much larger variety of media sources by working more cost efficiently and providing 24 hours coverage and availability.

This thesis investigates how professional media monitoring, which is currently a largely manual process, can be automatically supported. Three main modules are necessary for automatic media monitoring: speech recognition, topic segmentation, and topic classification. The research that was conducted on these three topics, and the resulting innovations are presented. The performance of the individual modules, as well as the complete system, is thoroughly investigated.

The focus of this thesis are German news. Topic boundaries are determined using a novel approach to visual indexing. A speech recogniser transforms the audio signals into texts, which are afterwards classified for the presence of pre-defined topics. For topic classification, approaches with Hidden Markov Models, Neural Networks, and Support Vector Machines (SVMs) are investigated. One contribution of this thesis is the introduction of novel couplers for SVMs with advantages over known couplers.

An additional topic covered in this thesis is Unsupervised Topic Discovery, a field nearly neglected in the literature. It makes it possible to find key-words in texts without a pre-defined topic list or training samples.

Contents

List of figures

List of tables

Chapter 1

Introduction

Information is of strategic importance for business and governmental agencies, but also for individual citizens. Nowadays, information is mainly obtained by manually analysing (reading, listening and watching) text resources, audio and video databases, and current broadcast multimedia sources (newspapers, magazines, radio, television, newswire, world wide web). Media monitoring companies play an important role in dissemination of information. They determine stories that are of interest to their customers, and notify them about relevant stories. Currently, the complete work of determining stories has to be carried out manually. Employees of media monitoring companies have thousands of profiles in mind that describe the kind of information customers are interested in.

The use of automatic methods for selection and dissemination of information would enable media monitoring companies to cover a much larger variety of media sources by working more cost efficiently and providing 24 hours coverage and availability.

1.1 Thesis outline

The subject of the presented thesis is a demonstrator of an automatic media monitoring system that automatically scans TV broadcast news for specific topics. It aims at assisting media monitoring companies with the monitoring process. Whenever topics of interest are automatically detected, the relevant customers can be alerted, e.g. by email. However, because an automatic system will never be completely reliable, the system's results will always have to be checked by a human. The system presented here is the first of its kind for German broadcast news. While much related research has been carried out on the level of components of the system, e.g. on automatic speech recognition or topic classification of broadcast news, those components were never before integrated into a system specifically designed for media monitoring of German news. It must be emphasised that the system is a demonstrator and not a final

product. Currently, it can only be applied to German news, because the speech recogniser only transcribes German.

This thesis deals both with the building blocks (modules) of the demonstrator, and with the system and its performance as a whole. Research and development on the building blocks has lead to several innovations, which will be presented in the respective chapters.

Figure 1.1 illustrates on a functional level how TV news broadcast documents (mpeg files), or plain, pre-segmented texts are processed by the system. Visual processing techniques are used to segment a news show into scenes, such as *newscaster* or *report*, and to detect story boundaries. The audio track is transcribed by an advanced automatic speech recogniser developed for transcription of broadcast news. The system is also capable of handling pure text data that was already segmented into stories (e.g. acquired from the Internet). In this case, the topic segmentation and speech recognition modules are bypassed, and the text is passed directly to the topic classification module. Otherwise, for mpeg news shows, automatic transcription and automatic topic segmentation is used for topic classification.

The topics found in one or several TV news shows, together with the transcription and information about the times of the story boundaries, are converted into XML format. The files can be accessed via a dedicated web browser interface that allows to search the transcriptions and the topic labels for specific words. The stories that match the search query are displayed line by line, together with the assigned topic, and a click on a story opens a window showing the transcription data and the meta-data, e.g. the topic label, start and end time of the story, and programme name. The core modules of the presented media monitoring system employ state-of-the-art stochastic classifiers.

The most important challenges to the topic recognition module are:

- The speech recogniser's transcription is not perfect. Especially with interviews or reports with background noise, it is sometimes hard even for humans to understand the contents from the transcription. With some stories, it is necessary to listen to the audio to grasp the contents.

- The goal of the system is to find topic categories in spoken documents (audio or video files with speech content). Stochastic classifiers rely on the fact that the properties of the training data match the test data. This condition is not fully met, since only summaries were available as training data, i.e. the training data and the test data stem from different domains.

- The majority of the stories in a news show are not related to any media monitoring customer, but are off-topic. However, off-topic training data is not available.

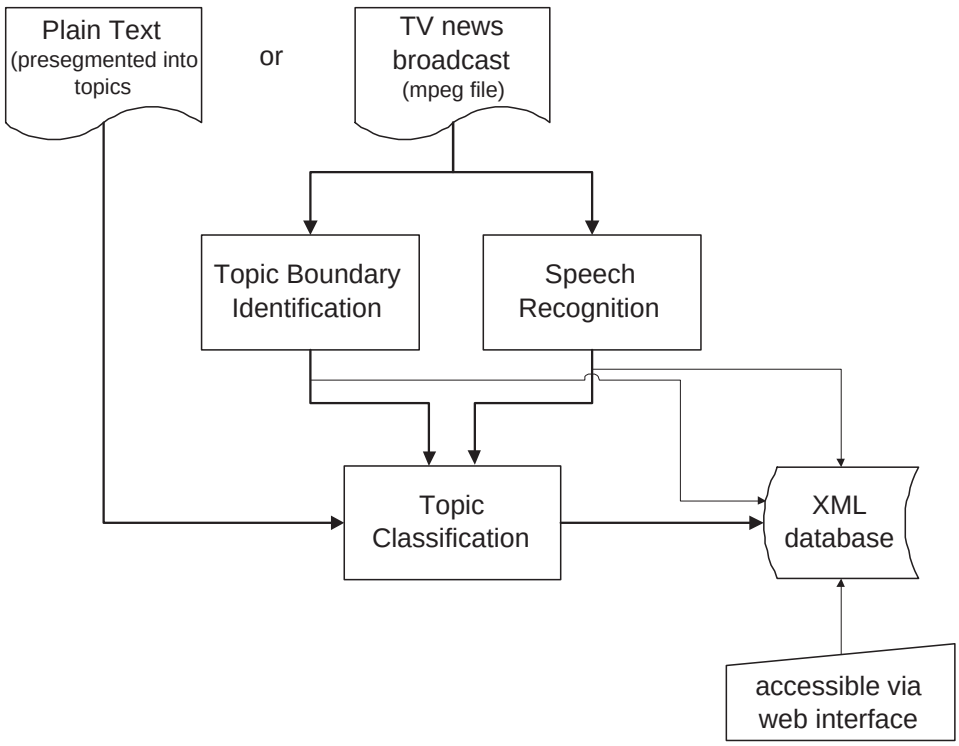


Figure 1.1: Functional structure of the automatic media monitoring demonstrator. The main workflow paths are printed in bold.

- The topics change rapidly over time. As the set of topics to be detected is assumed to be fixed (see below), there are topics in the test sets that are not present in the training set, and therefore cannot be detected.
- The final test set used across all experimental chapters of this thesis is especially challenging (see below for details).

Two simplifying assumptions were made for this thesis:

- The set of topics is assumed to be fixed.
- It is assumed that there is only one topic per story. This assumption is nearly true for the tested news shows; only two stories are annotated with more than one reference topic.

These assumptions were given up for the Unsupervised Topic Discovery task described in Chapter 10.

The outline of this thesis is as follows. Chapters 2 and 3 lay the theoretical foundations for the two most important implemented classification algorithms, Hidden Markov Models and Support Vector Machines. Before presenting approaches and experiments for the individual modules, Chapter 4 introduces the measures that are used to evaluate the performance both

of the modules and of the whole system. The evolution from a generic speech recogniser to a broadcast speech recogniser is presented in Chapter 5. Chapter 6 deals with topic segmentation of a TV news show. This segmentation aims at providing topically homogeneous stories. The various approaches taken for the topic classification modules are covered in the following two Chapters. Chapter 7 presents the methods based on Hidden Markov Models, and Chapter 8 treats the Support Vector Machine classifier. To given an impression of the performance of the combination of all modules, the overall system performance is measured in Chapter 9. A private user normally does not want to explicitly specify his or her interests by means of providing topics or profiles, but instead prefers to quickly browse news shows for interesting stories. Chapter 10 extends the domain of the application from professional media monitoring to "monitoring for everybody". The final chapter concludes this thesis, summarises important contributions, and gives an outlook. The storage of the monitoring results in an XML database and the access to it are not covered in this thesis.

It should be noted that the terms *document*, *report* and *story* are used interchangeably within this work, since the automatic transcription of TV news stories (also called news reports) is stored in text format. Thus, the topic classifier deals with text documents, regardless of whether they were produced by a speech recogniser or whether they were originally plain texts.

1.2 Data sets

With a single exception, the data sets that were used for training and testing the system and its components are in German. Two sources are predominant: manually created summaries of TV and radio reports (not restricted to *news* reports), and news shows from the two most important German TV channels. The evaluations of the approaches covered in this thesis are often done with one (or more) preliminary test sets, and also with a final test set (the SVM classifier was evaluated with a final test set only). The preliminary sets differ, but the final test data always consists of German TV news shows covering a period of two weeks: October 8th until 21st, 2001 [1] (the evaluation of the speech recogniser uses data only from the second week). This period is just four weeks after the attack of September 11th, 2001, and thus many reports are about this attack and the actions taken in its aftermath (for example, the war in Afghanistan). The result is that for some news shows, a great portion of the reports is about the same topic. Stories are therefore much longer than they would usually be. It has to be emphasised that the test set consists of "real life" data which reflect the requirements of a media monitoring company.

[1] This period was defined by the European Union project ALERT to be used for its common test set.

Chapter 2

Hidden Markov Models

Hidden Markov Models (HMMs) are very popular and successful classifiers widely used for recognition of speech [38], handwriting [71] and gesture [96]. A Hidden Markov Model is a stochastic machine that consists of connected states (see Figure 2.1). Of its n states $S = \{s_1, s_2, \ldots, s_n\}$, only one is active at each time step. The active state at time t is denoted by $q_t \in S$ and depends on the preceding active state, q_{t-1}, only, not on more recent states (first order Markov model). The sequence of visited states is denoted by $Q = q_1, q_2, \ldots, q_T$. The transition from state q_{t-1} to q_t occurs with a given probability:

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i). \tag{2.1}$$

All $n \times n$ transition probabilities are stored in the transition probability matrix $\mathbf{A} = [a_{ij}]$. The probability of state j being the initial state is

$$\pi_j = P(q_1 = s_j), \qquad \sum_{j=1}^{n} \pi_j = 1, \qquad \boldsymbol{\pi} = [\pi_j]. \tag{2.2}$$

Usually, it is assumed that a sequence of active states always begins at the first state

$$\boldsymbol{\pi} = (1, 0, 0, \ldots)^T \tag{2.3}$$

and ends at the last state. Each time a state is reached, an output vector $\mathbf{o}$ is emitted with a probability that depends on the active state only. $\mathbf{o}$ may be a continuous vector (continuous HMMs), or a single discrete number (discrete HMMs).

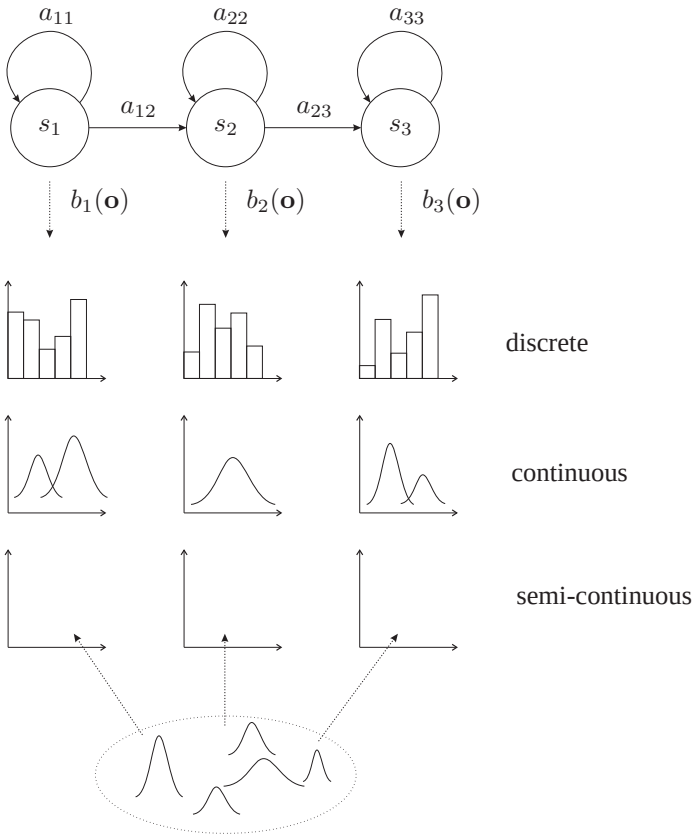


Figure 2.1: HMM with different types of emission probabilities.

Continuous HMMs In the case of *continuous HMMs*, the probability density function describing the emission probability of an output vector $\mathbf{o}$ is usually a weighted sum of multivariate Gaussian distributions $\mathcal{N}(\cdot)$:

$$b_j(\mathbf{o}) = p(\mathbf{o}|s_j) = \sum_{k=1}^{K} c_{k\,j} \cdot \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_{k\,j}, \boldsymbol{\Sigma}_{k\,j}), \tag{2.4}$$

$$\mathcal{N}(\mathbf{o}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^M |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^{\mathrm{T}}\,\boldsymbol{\Sigma}^{-1}\,(\mathbf{o}-\boldsymbol{\mu})}. \tag{2.5}$$

The index j denotes the state, k indicates the mixture component. $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix of the Gaussian distribution $\mathcal{N}(\cdot)$. M is the dimension of the observation vector $\mathbf{o}$. The emission distributions satisfy the probability constraints:

$$\int_{\mathbf{o}} b_j(\mathbf{o})\ \mathrm{d}\mathbf{o} = 1, \qquad b_j(\mathbf{o}) \in [0, 1]. \tag{2.6}$$

The mixture weights fulfil

$$c_{kj} \geq 0, \qquad \sum_{k=1}^{K} c_{kj} = 1. \tag{2.7}$$

Semi-continuous HMMs Semi-continuous HMMs are very similar to continuous HMMs. The only difference is that with semi-continuous HMMs the Gaussian components of the emission probability are not estimated separately for each state. Instead, a common codebook of L Gaussian mixtures is used:

$$b_j(\mathbf{o}) = \sum_{l=1}^{L} c_{l\,j} \cdot \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l). \tag{2.8}$$

Discrete HMMs One common technique is to map continuous observation vectors to discrete symbols by means of a vector quantisation step. Each observation is being replaced with its nearest prototype vector which is chosen from a pre-defined codebook of J prototype vectors. As the number of prototypes is fixed and known a-priori, it is sufficient to represent each prototype by its index m, $1 \leq m \leq J$:

$$\mathbf{o} \rightarrow m. \tag{2.9}$$

See Section 7.3.2 for details on vector quantisation. *Discrete HMMs* model a stream of discrete indices m, as opposed to continuous HMMs which model a stream of continuous observations $\mathbf{o}$. Consequently, the emission probabilities are also discrete:

$$b_j(\mathbf{o}) = b_j(m) = P(m|s_j), \qquad \sum_{m=1}^{J} b_j(m) = 1. \tag{2.10}$$

Let the emission probabilities of all states j of a HMM be represented by the symbol $\mathbf{B}$. For continuous HMMs, $\mathbf{B}$ contains – for each state j – the mean vectors and the covariance matrices of the Gaussian distributions, and the mixture weights: $\mathbf{B} = \langle b_j(\mathbf{o}) \rangle$. In the case of discrete HMMs, $\mathbf{B}$ is a $n \times J$ matrix: $\mathbf{B} = [b_j(m)]$.

A Hidden Markov Model λ is defined by the parameters $\boldsymbol{\pi}$, $\mathbf{A}$ and $\mathbf{B}$:

$$\lambda = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}). \tag{2.11}$$

2.1 Production probabilities

One important figure in the context of HMMs is the production probability $P(\mathbf{O}|\lambda)$, or the probability that an observation sequence $\mathbf{O} = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$ was generated by the Hidden Markov Model λ. The production probability plays an important role for predicting from which model λ a given observation sequence $\mathbf{O}$ was created, i.e. which class it belongs to most probably. In other words, it is used for the classification with HMMs.

As it is not known which state sequence $Q = (q_1, \ldots, q_T)$ has produced $\mathbf{O}$ (this is why one talks about *Hidden* Markov Models), the production probability is the result of a summation over all possible state sequences, i.e. all possible ways through the HMM are considered:

$$P(\mathbf{O}|\lambda) = \sum_{Q} P(\mathbf{O}, Q|\lambda). \tag{2.12}$$

The summed term is the probability that a given HMM λ goes through the state sequence $Q = (q_1, \ldots, q_T)$ and emits $\mathbf{O}$. The production probability can thus be expressed as [64]

$$P(\mathbf{O}|\lambda) = \sum_{Q} \left(\pi_{q_1} b_{q_1}(\mathbf{o}_1) \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{o}_t) \right). \tag{2.13}$$

The run-time of this calculation grows exponentially with the length of the observation T, because n^T possible state sequences have to be calculated. A more efficient approach is the forward-backward procedure described in Section 2.2.

2.2 Forward-Backward algorithm

As already mentioned, the Forward-Backward (FB) algorithm provides an efficient calculation of the production probability $P(\mathbf{O}|\lambda)$. The FB procedure is the solution of the EM (Expectation-Maximisation) algorithm applied to HMMs. The EM algorithm [21] allows to iteratively estimate model parameters so that they maximise the likelihood (which is, in the case of HMMs, $P(\mathbf{O}|\lambda)$). It is used when data is hidden or missing that is relevant for the stochastic process (here: generation of observations with HMMs) [86]. From the reference labelling of the training data, it is only known which HMM has produced the data, but not which state inside the HMM. As the state sequence is hidden (hence the name *Hidden* Markov Models), the EM algorithm is an ideal candidate for estimation of λ. The missing data is characteristic of the EM algorithm; if no data were missing during training, a normal maximum

likelihood algorithm could be used, i.e. the $\hat{\lambda}$ that maximises the log-likelihood $l(\lambda)$ [23] could be found.

For continuous HMMs whose emission probabilities $b_j(\mathbf{o})$ are a mixture of Gaussian densities, the EM algorithm can also be applied to get the weights of the Gaussians: For a given observation $\mathbf{o}$, it is not known which of the Gaussians has contributed to which extend to $\mathbf{o}$, i.e. the weights are hidden. See [65] for details.

The main idea of the FB algorithm is to recursively calculate forward probabilities

$$\alpha_t(j) = P(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t, q_t = s_j|\lambda) \tag{2.14}$$

and backward probabilities

$$\beta_t(i) = P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \ldots, \mathbf{o}_T|q_t = s_i, \lambda). \tag{2.15}$$

The calculation of $\alpha_t(j)\ \forall t, j$ (similarly, $\beta_t(i)\ \forall t, i$) leads to $T \times n$ matrices. The forward probability $\alpha_t(j)$ indicates the probability to observe the vector sequence $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t$ and to be, at time t, at state s_j. $\alpha_t(j)$ is initialised at the first time step $t = 1$

$$\alpha_1(j) = \pi_j b_j(\mathbf{o}_1) \qquad \forall\, 1 \leq j \leq n, \tag{2.16}$$

and then recursively calculated from the previous time step $t - 1$

$$\alpha_t(j) = \left(\sum_{i=1}^{n} \alpha_{t-1}(i) a_{ij} \right) \cdot b_j(\mathbf{o}_t). \tag{2.17}$$

The recursion ends at time T.

The backward probability $\beta_t(i)$ indicates the probability to observe the vector sequence $\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \ldots, \mathbf{o}_T$, given that at time t the model was at state s_i. The backward algorithm's recursion proceeds backwards in time. β is initialised at the last time step T

$$\beta_T(j) = 1 \qquad \forall\, 1 \leq j \leq n \tag{2.18}$$

and computed from the following time step $t + 1$ as

$$\beta_t(i) = \sum_{j=1}^{n} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j) \qquad \forall\, 1 \leq i \leq n. \tag{2.19}$$

The production probability can be expressed as

$$P(\mathbf{O}|\lambda) = \alpha_T(n) \tag{2.20}$$

(assuming that the state sequence Q always ends at the last state s_n). Alternatively it can be calculated incorporating the backward probabilities:

$$P(\mathbf{O}|\lambda) = \sum_{j=1}^{n} \pi_j b_j(\mathbf{o}_1)\beta_1(j), \qquad \text{or} \tag{2.21}$$

$$= \sum_{j=1}^{n} \alpha_t(j)\beta_t(j). \tag{2.22}$$

The product of the forward and the backward probability yields the probability that $\mathbf{O}$ is observed while being in state j at time t

$$\alpha_t(j)\beta_t(j) = P(\mathbf{O}, q_t = s_j|\lambda). \tag{2.23}$$

The latter two equations play an important role for parameter estimation with the Baum-Welch algorithm (see Section 2.3.1).

The run-time of the Forward-Backward procedure grows by n^2T, and thus linearly with the length of the observation sequence.

Strictly speaking, the Forward-Backward algorithm as described above is only valid for continuous HMMs. However, it is straightforward to change it in such a way that it can be applied to discrete HMMs: the continuous observations and the emission probabilities have to be replaced by their discrete counterparts

$$\begin{aligned} \mathbf{o}_t &\rightarrow m_t \\ \mathbf{O} = (\mathbf{o}_1, \ldots, \mathbf{o}_T) &\rightarrow \mathbf{O} = (m_1, \ldots, m_T) \\ b_j(\mathbf{o}_t) &\rightarrow b_j(m_t). \end{aligned} \tag{2.24}$$

2.3 Training

Before HMMs can be used to classify unknown observations, their parameters λ have to be set. The corresponding procedure is referred to as *training*, and it is done by estimating an optimal λ from training observation sequences $\mathbf{O}$. The optimisation criterion of the training is

the maximisation of the likelihood $P(\mathbf{O}|\lambda)$, i.e. those parameters $\hat{\lambda}$ are chosen that maximise the likelihood:

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} P(\mathbf{O}|\lambda). \tag{2.25}$$

Two methods are known that implement this maximum-likelihood estimation: the Baum-Welch algorithm, and the Viterbi training. Both will be described in the following sections.

2.3.1 Baum-Welch algorithm

The Baum-Welch algorithm [9] estimates the HMM parameters iteratively according to the EM (Expectation-Maximisation) algorithm [21]. Here, we derive the Baum-Welch algorithm for discrete HMMs. In each iteration step, new model parameters λ^* are estimated from the parameters λ of the model from the previous iteration step. It is guaranteed that the likelihood always increases between two iterations, until a local maximum is reached:

$$P(\mathbf{O}|\lambda^*) \geq P(\mathbf{O}|\lambda). \tag{2.26}$$

Let $\xi_t(i,j)$ be the a-posteriori probability of a state change from s_i to s_j. With (2.22) and (2.23), $\xi_t(i,j)$ becomes

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j|\mathbf{O},\lambda) = \frac{P(q_t = s_i, q_{t+1} = s_j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \tag{2.27}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{n}\alpha_t(i)\beta_t(i)}. \tag{2.28}$$

Let $\gamma_t(j)$ be the a-posteriori state probability $\gamma_t(j) = P(q_t = s_j|\mathbf{O},\lambda)$. Again, with (2.22) and (2.23), $\gamma_t(j)$ becomes

$$\gamma_t(j) = P(q_t = s_j|\mathbf{O},\lambda) = \frac{P(q_t = s_j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{n}\alpha_t(i)\beta_t(i)}. \tag{2.29}$$

Moreover,

$$\gamma_t(i) = \sum_{j=1}^{n}\xi_t(i,j). \tag{2.30}$$

With these definitions, the new parameters λ^* of *discrete* HMMs are estimated as [66]

$$\pi_i^* = \gamma_1(i) \tag{2.31}$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.32}$$

$$b_{jm}^* = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot \delta(\mathbf{o}_t = m)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{2.33}$$

The Kronecker function $\delta(\cdot)$ yields 1 if its argument is true, and 0 otherwise. The formula for a_{ij}^* is the ratio of the expected number of transitions from state s_i to state s_j, divided by the expected number of transitions out of state s_i. b_{jm}^* is the ratio of the expected number of times of being in state s_j and observing symbol m, divided by the expected number of times of being in state s_j. The formulae for *continuous* HMMs are similar and can be found in e.g. [64]. The training of the HMM parameters can alternatively be performed with the Viterbi algorithm presented in Section 2.4.2.

2.4 Classification

2.4.1 General approach

HMM classification inversely applies first-order Markov Models. In order to classify an observation sequence $\mathbf{O}$ into one of K classes, it is assumed that $\mathbf{O}$ was produced by a Hidden Markov Model. Each class y_i, $1 \leq i \leq K$ is modelled by one HMM λ_i whose parameters were already found (for example, by the Forward-Backward algorithm). The class of $\mathbf{O}$ is predicted by finding the HMM λ_i that has generated $\mathbf{O}$ with the highest posterior probability (maximum a-posteriori classification):

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, P(y_i|\mathbf{O}) = \underset{i}{\operatorname{argmax}}\, P(\lambda_i|\mathbf{O}). \tag{2.34}$$

With the Bayes formula, this equation turns into

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \frac{P(\mathbf{O}|\lambda_i)P(\lambda_i)}{P(\mathbf{O})} = \underset{i}{\operatorname{argmax}}\, P(\mathbf{O}|\lambda_i)P(\lambda_i) \tag{2.35}$$

since $P(\mathbf{O})$ is independent of the class. $P(\mathbf{O}|\lambda_i)$ is calculated from HMM λ_i using the Forward-Backward algorithm (Section 2.2) or the Viterbi algorithm.

2.4.2 Viterbi algorithm

Calculating the production probability $P(\mathbf{O}|\lambda)$ according to (2.12), i.e. by going through all possible state sequences, is prohibitively time consuming. The Viterbi algorithm [89] approximates $P(\mathbf{O}|\lambda)$ by considering only the state sequence Q^* that contributes most to the sum in (2.12)

$$P^*(\mathbf{O}|\lambda) = \max_Q P(\mathbf{O}, Q|\lambda) = P(\mathbf{O}, Q^*|\lambda). \tag{2.36}$$

That is, the most probable state sequence Q^* that has produced $\mathbf{O}$ is found. Eq. (2.36) is obtained similarly to the Forward-Backward algorithm using (2.20). The only difference is that the sum in the iteration of the forward probability (2.17) is replaced by the maximum operator. In each iteration step, the state which produced the maximum forward probability can be saved in order to obtain the most probable state sequence Q^*.

Chapter 3

Support Vector Machines

An important classifier paradigm next to Hidden Markov Models are Support Vector Machines (SVMs). Both are supervised learners, that is, their respective parameters are estimated from a training set. The class of each training example is given[1]. In contrast to generative classifiers, which model the distribution of the data of every class, SVMs learn only the decision boundary between classes, and are an example of discriminative classifiers. While multi-class problems can be straightforwardly implemented with HMMs, the basic form of SVMs can only separate two classes. They are not able to model dynamic data in the same way as HMMs, but they require less training data (because they are discriminative) and less parameter tuning. Another difference is that SVMs classify one feature vector, whereas HMMs classify a sequence of feature vectors.

SVMs date back to 1992 [13] and have proven to be very efficient for a wide variety of classification problems, such as document classification [43], pedestrian detection [59], handwriting recognition [8], gene classification [14], and many others.

3.1 Linear hard-margin SVMs

In their basic form, Support Vector Machines are linear, binary classifiers, that is, they find a hyperplane that optimally separates two classes.

[1] This is different to unsupervised training, where class labels are not available.

3.1.1 The optimal hyperplane

For a given set of linearly separable data points $\mathbf{x}_i \in \mathbb{R}^M$ and corresponding class labels $y_i \in \{-1;\ +1\}$, a plane that separates the two classes $+1$ and -1 can be expressed by the equation

$$\mathbf{w} \cdot \mathbf{x} + b = 0. \tag{3.1}$$

All points $\mathbf{x}$ that satisfy this equation lie on the plane. The normal vector of the plane, $\mathbf{w} \in \mathbb{R}^M$, is often called weight vector in the context of SVMs. The plane's distance from the origin of the coordinate system equals $\frac{|b|}{\|\mathbf{w}\|}$, with $b \in \mathbb{R}$.

The discriminating hyperplane is fully characterised by the set $\{\mathbf{w}, b\}$.

All data points that belong to class $y_i = 1$ should lie on one side of the plane, while all $\{\mathbf{x}_i | y_i = -1\}$ should lie on the other side. Without loss of generality, the normal vector $\mathbf{w}$ can be oriented in such a way that

$$\mathbf{w} \cdot \mathbf{x}_i + b > 0 \qquad \text{for } y_i = 1 \tag{3.2}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b < 0 \qquad \text{for } y_i = -1 \tag{3.3}$$

or

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0 \tag{3.4}$$

If this inequality is fulfilled for all i, all data points $\{\mathbf{x}_i\}$ can be separated without error.

For a given linearly separable data set, there are many possible discriminating hyperplanes. SVMs find the optimal hyperplane, which is the one that separates the data with the maximum margin. The margin δ is the distance between the hyperplane and the closest data point from every class[2]. It is the same for both classes, since otherwise the plane would be nearer to one class than to the other. Those training data points whose distance to the separation hyperplane is δ are called *support vectors* (see Figure 3.1). The two planes in parallel to the separation hyperplane that touch the support vectors $\mathbf{x}^{(s)}$ of either class fulfil

$$\mathbf{w} \cdot \mathbf{x}^{(s)} + b = \pm c \tag{3.5}$$

[2]This statement is only true for hard-margin decision functions, but not for soft margins.

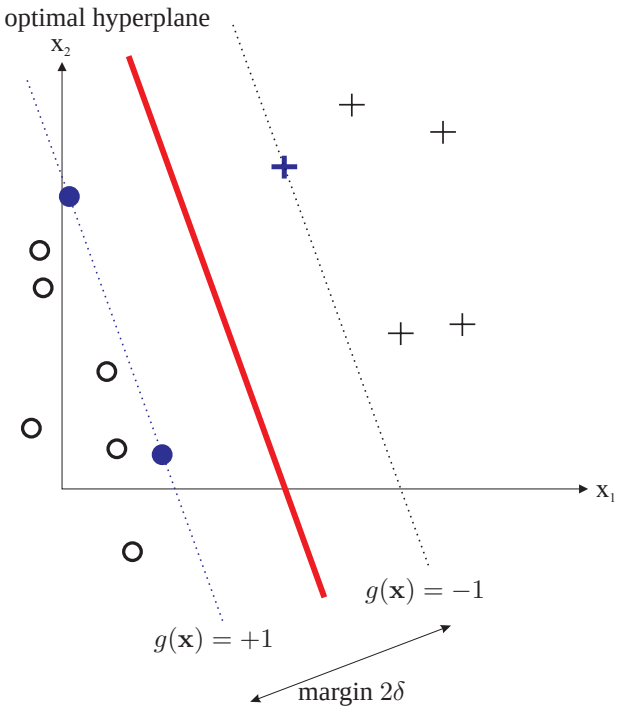


Figure 3.1: Hard margin hyperplane. Vectors in bold are support vectors.

with c a constant. $\mathbf{w}$ and b can be scaled in such a way that $c = 1$, without affecting the orientation of the separating plane:

$$\mathbf{w} \cdot \mathbf{x}^{(s)} + b = \pm 1. \tag{3.6}$$

The distance of a point $\mathbf{x}$ to the hyperplane is [30, p. 322]

$$\frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|}, \tag{3.7}$$

so the discriminating function

$$g(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.8}$$

is a measure of this distance and can therefore be used as a measure of confidence of the classification. Its sign tells on which side of the separating hyperplane a data point $\mathbf{x}$ lies (compare (3.2) and (3.3)). Consequently, the decision function $h(\mathbf{x}) \in \{-1, +1\}$ of the linear SVM, which hypothesises the class of a test data point $\mathbf{x}$, is

$$\hat{y} = h(\mathbf{x}) = \text{sign}(g(\mathbf{x})) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b). \tag{3.9}$$

From (3.6), (3.7) and (3.8), the distance of any support vector $\mathbf{x}^{(s)}$ to the optimal hyperplane is

$$\delta = \frac{|g(\mathbf{x}^{(s)})|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}. \tag{3.10}$$

The margin of separation between the two classes is thus 2δ.

3.1.2 Calculating the optimal hyperplane

The position of the separating hyperplane is determined during the training phase of the SVMs by solving an optimisation problem. Maximising the margin $2\delta = \frac{2}{\|\mathbf{w}\|}$ is equivalent to minimising $\frac{1}{2}\|\mathbf{w}\|$, or minimising $\frac{1}{2}\|\mathbf{w}\|^2$, since $(\cdot)^2$ is monotonic increasing. To find the parameters $\{\mathbf{w}, b\}$ of the optimal hyperplane, one has to solve the following primal optimisation problem:

- Minimise

$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}\mathbf{w} \tag{3.11}$$

- subject to zero training error

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad \text{for } i = 1, \ldots, N \tag{3.12}$$

N is the number of labelled examples $\{\mathbf{x}_i, y_i\}$ used for training. Note that a zero training error is also described by

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0, \tag{3.13}$$

but if this inequality is used as the constraint it is not possible to find a minimum of $\frac{1}{2}\|\mathbf{w}\|$: The solution of the optimisation problem, $\{\mathbf{w}, b\}$, can be rescaled with some $0 < \lambda < 1$ without changing the optimal hyperplane. This still satisfies the constraint (3.13) (because $\lambda > 0$), but the length of $\mathbf{w}$ has decreased. Hence, the solution $\{\mathbf{w}, b\}$ is not the minimal one. Due to rescaling, $\|\mathbf{w}\|$ can be made arbitrarily small. Using constraint (3.12) limits rescaling and makes it possible to find a minimum value for $\|\mathbf{w}\|$ [77]. At the same time, this constraint guarantees that there are no training points inside the margin (compare eq. (3.6)).

The optimisation problem can be solved by introducing Lagrange multipliers $\alpha_i \in \mathbb{R}, \alpha_i \geq 0$ and constructing the Lagrangian L

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right) \tag{3.14}$$

L has to be minimised with respect to the primal variables $\mathbf{w}$ and b, and to be maximised with respect to the dual variables α_i [75]:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \qquad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0. \tag{3.15}$$

This leads to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.16}$$

and

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i. \tag{3.17}$$

The solution vector $\mathbf{w}$ is thus a linear combination of the training examples. For those training points that do not match the equality of (3.12), i.e. for which $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 > 0$, the corresponding α_i in (3.14) must be 0; otherwise, the term $\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right)$ would make $L(\mathbf{w}, b, \boldsymbol{\alpha})$ smaller, and thus prevent L from being maximised with respect to $\boldsymbol{\alpha}$. Only the training points that satisfy $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$ have a corresponding $\alpha_i > 0$; these training points are the support vectors (Kuhn-Tucker theorem of optimisation theory; compare (3.6)). This means that only the support vectors contribute to the orientation of the optimal hyperplane, all other training examples can be removed from the training set without affecting the solution.

Substituting (3.16) and (3.17) into the Lagrangian (3.14) leads to the following dual formulation of the optimisation problem:

- Find the Lagrange multipliers α_i that maximise

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{3.18}$$

- subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

This dual formulation has got the same solutions as the primal optimisation problem (3.11) and (3.12). Its solution is characterised by the Lagrange multipliers α_i, and not by the weight vector $\mathbf{w}$ and the bias b as in the primal formulation. Optionally, $\mathbf{w}$ can still be calculated using (3.17). To obtain the bias b, one support vector $\mathbf{x}^{(s)}$ must be arbitrarily selected and inserted into $y_i(\mathbf{w} \cdot \mathbf{x}^{(s)} + b) = 1$.

Moreover, the objective function $W(\boldsymbol{\alpha})$ depends only on the labelled training data in the form of a set of dot products. This fact makes it possible to classify patterns that are not linearly separable (see Section 3.2).

The expansion (3.17) is used to write the decision function in terms of the labelled training sample pairs $\{\mathbf{x}_i, y_i\}$ and the solution coefficients of the optimisation problem, α_i and b:

$$\hat{y} = h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b\right). \tag{3.19}$$

3.2 Kernels

The Support Vector Machine as it was presented this far is unable to separate non-linearly-separable patterns. Cover's theorem on the separability of patterns [30] states that if such patterns are transformed to a higher-dimensional space, they become linearly separable with high probability. Two conditions have to be met: First, the transformation must be non-linear. Second, the dimensionality of the new space must be high enough. An example is depicted in Figure 3.2.

The non-linear mapping function is denoted by $\Phi(\mathbf{x}) : \mathbb{R}^M \rightarrow \mathbb{R}^L, L \gg M$. The high-dimensional space $\mathbb{R}^L$ is often referred to as the *feature space*. To clearly distinguish the original space where the features reside from the space where the features are mapped to, this thesis does not strictly adhere to this convention, but rather calls $\mathbb{R}^L$ the *mapped feature space*.

The mapping function $\Phi(\mathbf{x})$ replaces all occurrences of $\mathbf{x}$ in the derivation of the linear SVM. The decision function then becomes

$$\hat{y} = h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i) + b\right). \tag{3.20}$$

As the dimension of the mapped features may be very large or even infinite, the costs of computing the dot (or inner) product $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)$ may become very large. However, one is

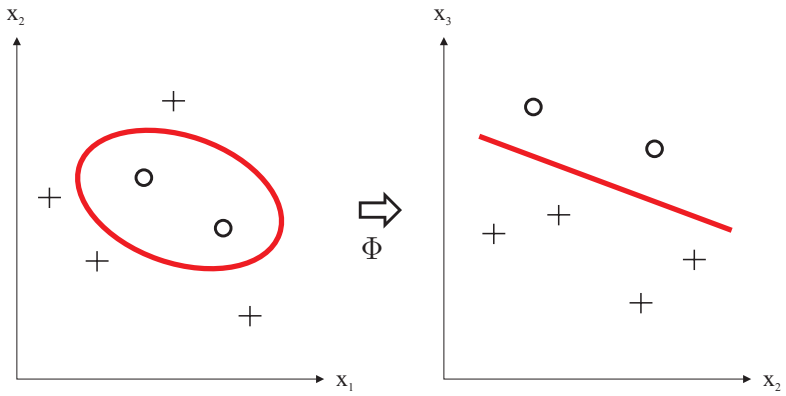


Figure 3.2: A non-linear mapping $\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$; $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ can make the input data in $\mathbb{R}^2$ linearly separable in $\mathbb{R}^3$. The original data is depicted on the left; after applying Φ the data becomes linearly separable (right). The hyperplane on the left is the result of projecting back the hyperplane from the high-dimensional space (right) into the original data space.

not interested in the mapped features themselves, but in the real-valued dot product. For some mapping functions Φ, the dot product can be directly calculated without performing the feature mapping by using a so-called kernel function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j). \tag{3.21}$$

The objective function $W(\boldsymbol{\alpha})$ of the dual optimisation problem (3.18) then becomes

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j). \tag{3.22}$$

The decision function (3.20) can be expressed as

$$\hat{y} = h(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b \right). \tag{3.23}$$

SVMs that do not make use of kernel functions are called linear SVMs; non-linear SVMs, on the other hand, use kernels. Table 3.1 lists some frequently used kernels together with their parameters. The correct choice of a kernel, as well as the correct value of the parameter(s) of a kernel must be determined using model selection methods like cross validation (see Section 8.4) or bootstrapping [23, 29]. Valid kernels, i.e. kernels that satisfy equation (3.21), have to be constructed according to Mercer's theorem [30, 76].

When a kernel is used, the vector $\mathbf{w}$ is the normal surface vector of the separating hyperplane

in the mapped feature space. The boundary that separates the classes is linear in the mapped feature space, but has non-linear shape in the original space. It cannot be calculated from the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ alone, because the mapping Φ must be known (which usually is not the case): with mapping, (3.17) becomes

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i). \tag{3.24}$$

The vector $\mathbf{w}$ is fortunately not necessary for classification (see (3.23)), because the dual formulation of the optimisation problem (3.22) is solved. The length of $\mathbf{w}$, however, can be directly calculated from the kernel:

$$\begin{aligned} \|\mathbf{w}\|^2 = \mathbf{w} \cdot \mathbf{w} &= \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i) \sum_{j=1}^{N} \alpha_j y_j \Phi(\mathbf{x}_j) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_j) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{3.25}$$

To put it in other words, $\mathbf{w}$ can be calculated only for linear SVMs, but the length of $\mathbf{w}$ can be calculated for both linear and non-linear SVMs.

Table 3.1: Some inner-product kernels for SVMs.

Kernel name	$k(\mathbf{x}_i, \mathbf{x}_j) =$	Parameters of the kernel
Linear	$\mathbf{x}_i \cdot \mathbf{x}_j$	—
Radial-basis function (RBF)	$\exp(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|)$	σ
Polynomial	$(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + c_0)^p$	$\gamma, c_0, p($ degree$)$
Sigmoid	$\tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + \beta)$	β, γ [a]

[a] Mercer's theorem is satisfied only for some β, γ [30].

3.3 Soft-margin decision functions

One way for SVMs to deal with non-separable training points is to allow for some points to be on the wrong side of the separating hyperplane, i.e. to allow data points $\mathbf{x}_i$ that violate the zero training error condition (eq. (3.12))

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad \text{for } i = 1, \ldots, N. \tag{3.26}$$

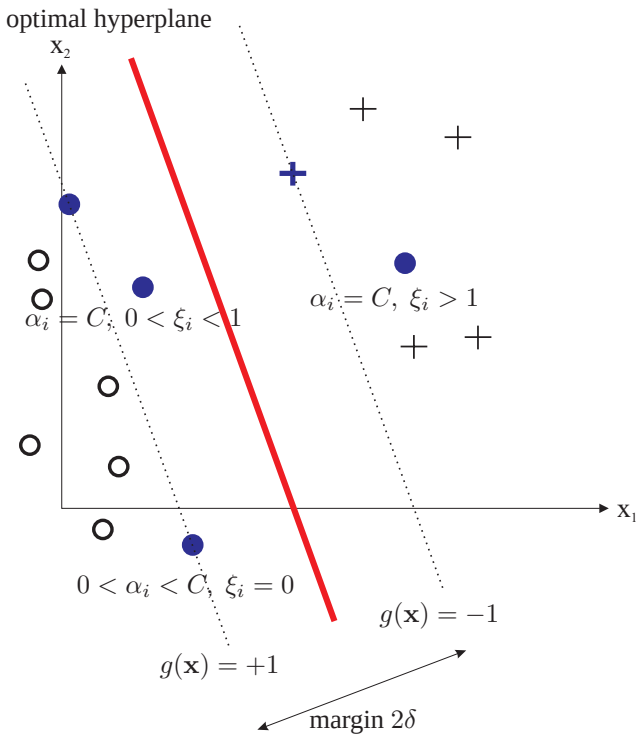


Figure 3.3: Soft margin hyperplane. Vectors in bold are support vectors; the corresponding values of ξ_i and α_i are stated.

In this case, the margin between the two classes is called a *soft margin*. Two types of violations may occur: (a) the data point lies on the right side of the hyperplane, but inside the margin (b) the data point lies on the wrong side of the hyperplane. In the second case, the data point would be misclassified by the SVM.

In order to allow violating data points, the constraint (3.12) is relaxed by introducing slack variables for every training point, $\xi_i \in \mathbb{R}^{\geq 0}, i = 1, \ldots, N$:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, \ldots, N. \tag{3.27}$$

For $0 \leq \xi_i \leq 1$, the corresponding data point falls inside the margin, but on the correct side of the hyperplane. For $\xi_i > 1$, it falls on the wrong side (see Figures 3.3 and 3.4).

The additional costs resulting from non-separability are incorporated by an extra cost term to the objective function (3.11), $C \sum_i \xi_i$. This sum measures the misclassification rate and should therefore be minimised. C is a real, pre-defined constant that specifies the trade-off between the degree to which misclassified data should be tolerated and the complexity of the discriminating surface. A large C will result in a complex surface that can separate the training data very well (see Figure 3.4). As will be described in Section 3.6.2, a wide margin yields a

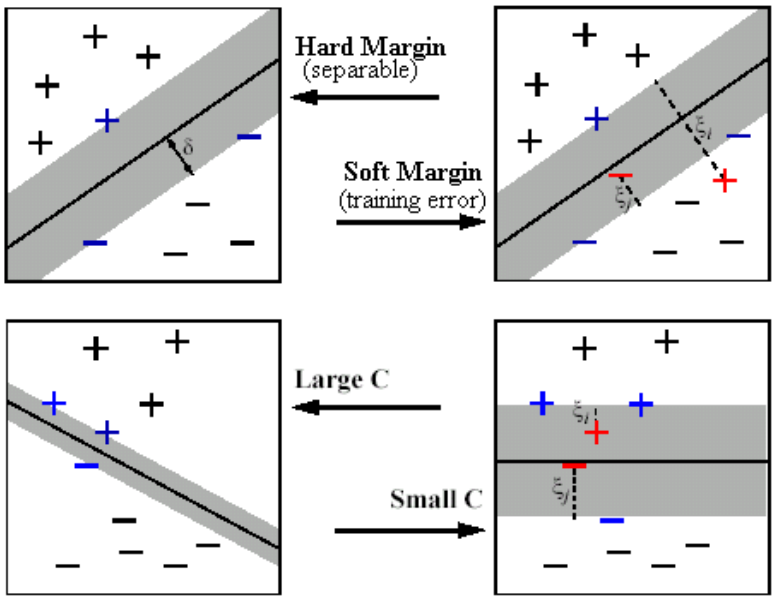


Figure 3.4: The influence of parameter C and slack variables ξ in soft margin SVMs.

less complex decision surface than a narrow one.

The optimisation problem that now has to be solved is:

- Find the optimal weight vector $\mathbf{w}$ and bias b such that $\mathbf{w}$ and the slack variables ξ_i minimise

$$\frac{1}{2}\mathbf{w}\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{3.28}$$

- subject to

$$y_i(\mathbf{w}\cdot\Phi(\mathbf{x}_i)+b) \geq 1-\xi_i \qquad \text{for } i=1,\ldots,N \tag{3.29}$$

It incorporates the – usually unknown – mapping function Φ. The corresponding dual formulation of the optimisation problem uses the kernel instead of Φ:

- Find the Lagrange multipliers that maximise

$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{3.30}$$

- subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.31}$$

$$0 \leq \alpha_i \leq C \qquad \text{for } i = 1, \ldots, N \tag{3.32}$$

It should be pointed out that this formulation is the same as for the case of linearly separable data, except for one detail: the α_i are no longer bounded at only one side ($\alpha_i \geq 0$), but at two sides: $0 \leq \alpha_i \leq C$. Those support vectors whose Lagrange multiplier satisfy $0 < \alpha_i < C$ must have $\xi_i = 0$, i.e. they lie exactly on the margin at a distance of $1/\left\|\mathbf{w}\right\|$ from the hyperplane. Values of $\alpha_i = C$ can only occur together with $\xi_i > 0$. These data points will either lie inside the margin, but be classified correctly ($\xi_i < 1$), or will lie on the wrong side of the hyperplane ($\xi_i > 1$) [91] (see Figure 3.3). Just as for hard margin SVMs, those training data points with $\alpha_i > 0$ are called support vectors. Hence, not only those points that lie exactly on the margin, but also those that lie inside the margin or on the wrong side of the hyperplane are support vectors. The latter are referred to as bounded support vectors. The dependence between the α_i and the ξ_i are a consequence of the Karush-Kuhn-Tucker conditions for optimality. Details can be found in e.g. [16] or [77].

3.4 Probabilistic SVMs

The Support Vector Machine as described above is only able to predict the class of a data point. Its decision value $g(\mathbf{x})$ can be used as a measure of how sure the prediction is. The drawback of $g(\mathbf{x})$ is that it cannot be compared across binary models, as will be described in Section 8.6.3. Moreover, it is not normalised or bounded, so a single decision value is difficult to interpret (except for the fact that $g(\mathbf{x}) = \pm 1$ means that the data point lies exactly on the margin).

These drawbacks can be addressed with probabilistic SVMs, which output the posterior probability that a test point $\mathbf{x}$ belongs to, say, class +1: $p(y = +1|\mathbf{x})$. The true value of the posterior is not known, but has to be estimated. One very popular approach to estimation was presented by Platt [62]. He argues that the posterior can be approximated using a sigmoid function with parameters A and B:

$$p(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(Ag(\mathbf{x}) + B)} \tag{3.33}$$

where $p(y = +1|\mathbf{x}) + p(y = -1|\mathbf{x}) = 1$. For estimation, A and B must be optimised using the decision values $g(\mathbf{x})$ of some training data points. Note that it does not make sense to estimate these parameters based on the distance $d(\mathbf{x})$ instead of the decision value, because one is a multiple of the other; the resulting posteriors will be the same.

Platt minimises the negative log likelihood of the training data using a model-trust algorithm. An improvement of his algorithm was presented by [49] and is used for the experiments reported on in this thesis. In order to distinguish between probabilistic and non-probabilistic (conventional) SVMs, the latter will from now on be referred to as non-probabilistic SVMs and abbreviated npSVMs.

The sigmoid function (3.33) (given that its parameters were estimated) is only applied during classification. First, the decision value $g(\mathbf{x})$ of an unknown test point $\mathbf{x}$ is computed with a conventional SVM as described in the previous sections (compare (3.23)):

$$g(\mathbf{x}) = \left(\sum_{i=1}^{N} y_i \alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b \right). \tag{3.34}$$

Then, (3.33) is used to get the posterior probability. Thus, pSVMs are just an extension of conventional SVMs.

The class prediction of a probabilistic SVM (pSVM) does not make use of the decision function $g(\mathbf{x})$, but of the posterior probability. It decides for class +1 if $p(y = +1|\mathbf{x}) > p(y = -1|\mathbf{x})$, otherwise it chooses class -1. Note that the decision boundary of non-probabilistic SVMs is at $g(\mathbf{x}) = 0$; the boundary for probabilistic SVMs is at

$$p(y = +1|\mathbf{x}) = p(y = -1|\mathbf{x}) = 0.5. \tag{3.35}$$

For $B \neq 0$, $p(y = \pm 1|g(\mathbf{x}) = 0) \neq 0.5$, i.e. the decision boundaries of conventional and probabilistic SVMs do not necessarily match.

Platt addresses the problem of the correct choice of the data set that is used to estimate the sigmoid parameters A and B. If the whole training set is used, some training points will be non-bounded support vectors (nbSV) and thus have a decision value of exactly $g(\mathbf{x}) = \pm 1$. The more nbSVs, the more values ± 1 will be used to estimate the sigmoid parameters, and so the more the parameters will be pushed (biased) against the margin. This will become a problem for (a) non-linear SVMs (SVMs with a non-linear kernel), where usually a substantial fraction of the training data will be nbSVs (b) linear SVMs, where the dimensionality of the input vectors is high compared to the number of training points (since there will be at maximum, $M + 1$ nbSVs for M-dimensional data). The latter case (b) was overlooked by Platt. He argues that for linear SVMs, especially with a small C, the bias is not severe. However, high-

dimensional data together with linearly separable classes play an important role in the context of text categorisation. Hence, avoiding a biased sigmoid is crucial for the work described in this thesis.

Several ways to avoid biased estimation are suggested by Platt. The best one is cross-validation of the training data: The whole training set is divided into, say, five equally sized portions. Four are used to train a SVM, i.e. to estimate a decision boundary between the two classes. From the remaining data, the decision values $g(\mathbf{x})$ are calculated using the just trained SVM. It is very unlikely that for this data $g(\mathbf{x}) = 1$, because it was not used to train the boundary. This procedure (training of SVM and calculation of $g(\mathbf{x})$) is repeated using the other four combinations of the data chunks. In the end, the whole training set has supplied decision values, which are then used to estimate the sigmoid parameters A and B. This 5-fold cross-validation method was (usually) used to create the experiments presented in this thesis.

3.5 Multi-class categorisation

The Support Vector Machine approach as described in the above sections can only distinguish two classes. However, many real-world problems are multi-class problems. Nevertheless, introduction books to SVMs usually just ignore multi-class categorisation, or address it only briefly.

Several approaches to deal with more than two classes are known:

- Direct modification of the SVM optimisation problem.
- **One-against-all**. The model for category $k = i$ is trained on all other $K - 1$ categories $k \neq i$, where K is the total number of categories ($1 \leq i \leq K$). For testing (predicting), the K decision values of the models, $g_k(\mathbf{x}), k \in [1, \ldots, K]$ are computed. The prediction result is the class that received the highest decision value:

$$\hat{y} \equiv \text{predicted class} = \underset{k}{\operatorname{argmax}}\, g_k(\mathbf{x}) \tag{3.36}$$

- **One-against-one**. For every combination of two disjoint classes, one SVM model is trained. I.e., $i - 1$ models are trained that contain class i ($\{i, 1\}, \{i, 2\}, \ldots, \{i, i - 1\}$). This amounts to a total of $K(K - 1)/2$ models. The combination of the same number of decision values in order to obtain one single prediction result is described in the following section.

- **Directed acyclic graph (DAG)**. As in the one-against-one approach, $K(K-1)/2$ models are trained. For testing, it uses a rooted binary directed acyclic graph that has $K(K-1)/2$ internal nodes and K leaves guiding the K classes. Figure 3.5 depicts a 4-class DAG.

Hsu and Lin [32] have compared these methods and conclude that the latter two methods are most suitable for practical use. Here, the one-against-one method is used, which is implemented in the `libsvm` software package [1].

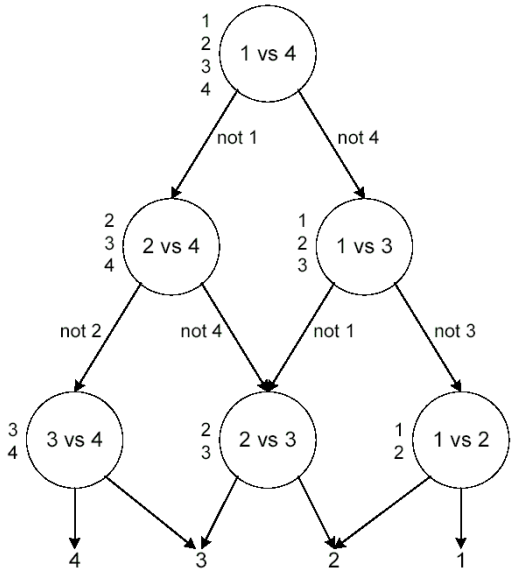


Figure 3.5: Directed Acyclic Graph for multi-class categorisation with SVMs, 4 class case.

3.5.1 Couplers for non-probabilistic SVMs

The step that combines the $K(K-1)/2$ distinct decisions of the one-against-one approach into one overall prediction is called *coupling*. This section discusses the standard coupler for non-probabilistic (conventional) SVMs, the voting coupler, and presents two new couplers. The following section deals with couplers for probabilistic SVMs.

The predominant *voting* coupler increments by 1 the voting counter of the winning class of every binary classifier. The class with the highest number of votes is the prediction result for the multi-class problem. In spite of its simplicity, the voting coupler works well. However, it does not incorporate any confidence measures of the binary classification, which reflect the certainty of the system that its prediction is correct.

A straightforward measure of confidence of a test vector $\mathbf{x}$ is the value of the decision function $g_{ij}(\mathbf{x})$, which is proportional to the distance $d_{ij}(\mathbf{x})$ of $\mathbf{x}$ to the separating hyperplane of the binary classifier (i, j): $d_{ij}(\mathbf{x}) = \frac{g_{ij}(\mathbf{x})}{\|\mathbf{w_{ij}}\|}$ ($\mathbf{w_{ij}}$ is the trained weight vector of the hyperplane). The index $_{ij}$ denotes the binary SVM model trained on the class pair (i, j). In order to test the influence of confidence, two new couplers, *decision value* and *distance*, were constructed where the decision values $g_{ij}(\mathbf{x})$ and the distances $d_{ij}(\mathbf{x})$ are added instead of 1. While $g_{ij}(\mathbf{x})$ was used as confidence measure in other contexts (error correcting output codes [68]), the author is not aware of publications that use the distance.

From a theoretical point of view, both the decision function $g_{ij}(\mathbf{x})$ and the distance $d_{ij}(\mathbf{x})$ have advantages: Because the decision function $g_{ij}(\mathbf{x}) = 1$ for data vectors $\mathbf{x}$ that lie on the margin, it yields confidence information scaled to the width of the margin. Assume a test vector with a fixed Euclidean distance to the separating hyperplane. If the margin is small, i. e. the classes can not be separated easily, the vector will lie outside the margin and return a high confidence value $g_{ij}(\mathbf{x}) > 1$. If, for the same vector, the margin is large and the two classes can be separated very well, the confidence value will be low. This behaviour makes sense because for easily separable classes, the test vector should lie far away from the hyperplane to get a high confidence measure. Thus, $g_{ij}(\mathbf{x})$ reflects how far away the test vector is from the margin. The advantage of the distance coupler is that the geometric distance $d_{ij}(\mathbf{x})$ obtained with one binary classifier (i, j) can be compared to the distance $d_{lm}(\mathbf{x})$ of another binary classifier (l, m), whereas the decision values can only be compared within *one* binary classifier (see Figure 3.6) [34].

3.5.2 Couplers for probabilistic SVMs

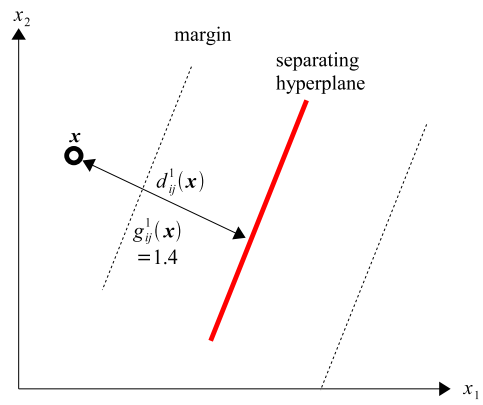
This section describes several couplers for probabilistic SVMs. In the following discussion, the following symbols will be used:

- p_i: overall a-posteriori class probability, $p_i = p(y = i|\mathbf{x})$.
- μ_{ij}: true pairwise a-posteriori class probability, $\mu_{ij} = p(y = i|\mathbf{x}, y = i \text{ or } j) = \frac{p_i}{p_i+p_j}$,
- r_{ij}: estimate of μ_{ij},

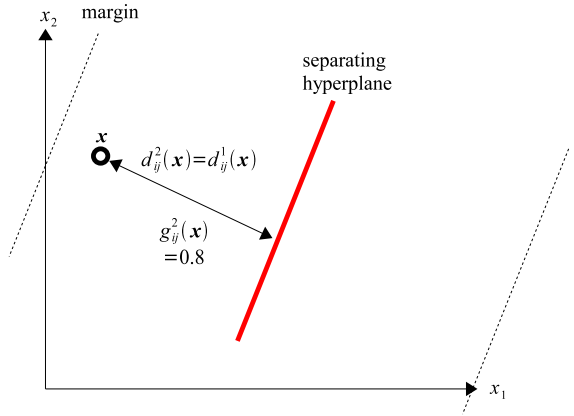
Couplers take the estimated pairwise a-posteriori probabilities that are output by the pSVMs (3.33) and transform them into overall a-posteriori probabilities[3]. Their class prediction output $\hat{y}$ is the class that received the highest a-posteriori probability:

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, p(y = i|\mathbf{x}). \tag{3.37}$$

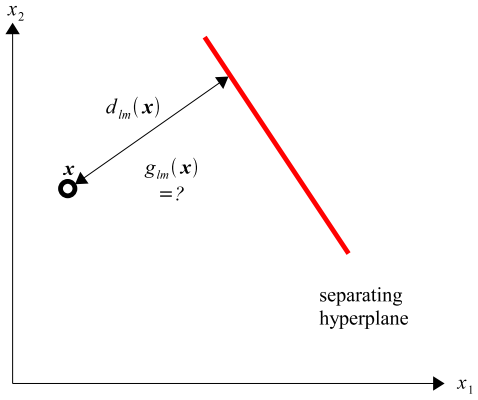
[3]The voting coupler does not directly calculate the overall posterior, but it can be derived from the votes.



(a) Classifier for classes i and j with small margin.



(b) Classifier for classes i and j with large margin.



(c) Classifier for classes l and m.

Figure 3.6: The distance $d(\mathbf{x})$ of a vector $\mathbf{x}$ to the separating hyperplane, and its decision value $g(\mathbf{x})$ are proportional to each other, but not equal. For a given hyperplane, $g(\mathbf{x})$ depends on the margin 2δ, but $d(\mathbf{x})$ does not. The decision value is relative to the margin (Figures (a) and (b)). When $\mathbf{x}$ is classified by two different binary classifiers (i, j) and (l, m), the (euclidean) distances $d_{ij}(\mathbf{x})$ and $d_{lm}(\mathbf{x})$ can directly be compared. But it is questionable whether the decision values $g_{ij}(\mathbf{x})$ and $g_{lm}(\mathbf{x})$ can be compared, since the margin of the two classifiers is (usually) different. This is indicated by the missing margin in Figure (c): Without knowledge of the margin, the decision value cannot be calculated.

The workflow of a coupler can be represented by

$$\{r_{ij}\}_{i,j=1}^{K} \quad \rightarrow \quad \{p_i\}_{i=1}^{K} \quad \rightarrow \quad \hat{y}. \tag{3.38}$$

Voting

The probabilistic voting coupler is analogous to the non-probabilistic coupler: for a test using the model of class pair (i, j) (where $1 \leq i, j \leq K$), it increments the voting counter $I(i)$ of the winning class i by one. The winning class is determined by comparing the pairwise posteriors:

$$\text{if } r_{ij} \begin{cases} > 0.5 & \text{then } I(i) \leftarrow I(i) + 1 \\ < 0.5 & \text{then } I(j) \leftarrow I(j) + 1 \end{cases} \qquad \forall i, j \tag{3.39}$$

The class with most votes is predicted:

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, I(i) \tag{3.40}$$

The overall (not pairwise) posterior probability of class i, $p_i \equiv p(y = i|\mathbf{x})$, can be estimated as the fraction of votes for i divided by the total number of votes:

$$p_i = 2I(i)/K(K-1) \tag{3.41}$$

The voting couplers for pSVMs and for npSVMs may lead to different class predictions, because the npSVM voting coupler makes a decision using the decision value $g(\mathbf{x})$, and the corresponding pSVM coupler uses the pairwise posterior probability. As was pointed out in Section 3.4, these two functions usually describe different class boundaries.

The following four sections describe couplers that are covered by and examined in Wu et. al. [95]. Among these couplers, two are already known couplers, and two are newly introduced by them.

Method by Price, Kner, Personnaz, and Dreyfus

The method by Price, Kner, Personnaz, and Dreyfus [63] considers that the sum of the overall posteriors over all classes is 1, and re-writes the posteriors:

$$1 = \sum_{j=1}^{K} p_j = (\sum_{j:j\neq i} p_i + p_j) - (K-2)p_i. \tag{3.42}$$

Using $r_{ij} \approx \frac{p_i}{p_i+p_j}$, one obtains the p_i,

$$\frac{1}{p_i} \approx \sum_{j:j\neq i} \frac{1}{r_{ij}} - (K-2), \tag{3.43}$$

which then have to be normalised so that $\sum p_i = 1$. This approach is referred to as PKPD.

Method by Hastie and Tibshirani

Hastie and Tibshirani [28] suggest minimising the Kullback-Leibler distance between r_{ij} and μ_{ij},

$$l(\mathbf{p}) = \sum_{i\neq j} n_{ij} r_{ij} \log \frac{r_{ij}}{\mu_{ij}}, \qquad \mathbf{p} = (p_1, \ldots, p_K)^T, \tag{3.44}$$

where n_{ij} is the number of training vectors in classes i or j. They present an algorithm to find the minimum ($\nabla l(\mathbf{p}) = 0$) using an optimisation algorithm. According to [95], it will find a unique global minimum. Instead of finding the true minimum $\mathbf{p}^*$, they find a vector $\tilde{\mathbf{p}}$ with

$$\tilde{p}_i = \frac{2\sum_{s:i\neq s} r_{is}}{K(K-1)} \tag{3.45}$$

whose elements are in the same order as those in $\mathbf{p}^*$ (i.e. $p_i^* > p_j^*$ iff $\tilde{p}_i > \tilde{p}_j$). It is sufficient to know $\tilde{\mathbf{p}}$ in order to perform classification. Using the identity

$$p_i = \sum_{j:j\neq i} \frac{p_i+p_j}{K-1} \cdot \frac{p_i}{p_i+p_j} = \sum_{j:j\neq i} \frac{p_i+p_j}{K-1} \mu_{ij} \tag{3.46}$$

and replacing $p_i + p_j$ with $2/K$ and μ_{ij} with r_{ij}, one obtains (3.45).

The coupler of Hastie and Tibshirani is abbreviated *HT*. One of its characteristics is that it requires an optimisation problem to be solved.

Markov coupler

The first new coupler presented by [95] is similar to the HT coupler. However, they do not replace $p_i + p_j$ by $2/K$, hence they solve the system

$$p_i = \sum_{j:j\neq i} \frac{p_i+p_j}{K-1} r_{ij} \tag{3.47}$$

subject to $\sum_{i=1}^{K} p_i = 1, p_i \geq 0$. This can be rewritten as

$$Q\mathbf{p} = \mathbf{p}, \quad \sum_{i=1}^{K} p_i = 1, \quad Q_{ij} = \begin{cases} r_{ij}/(K-1) & \text{if } i \neq j \\ \sum_{s:s\neq i} r_{is}/(K-1) & \text{if } i = j \end{cases} . \tag{3.48}$$

There is a finite Markov Chain whose transition matrix is Q, hence this algorithm is called a *Markov* coupler. The optimal $\mathbf{p}$ can be obtained by solving the linear system (3.48) using Gaussian elimination.

Minpair coupler

The minpair coupler, again a coupler first introduced by [95], optimises

$$\frac{1}{2}\sum_{i=1} K \sum_{j:j\neq i} (r_{ji}p_i - r_{ij}p_j)^2 \tag{3.49}$$

subject to $\sum_{i_1}^{K} p_i = 1, p_i \geq 0$. Wu et al. [95] show that this problem can be re-written as a linear system and solved with standard methods like Gaussian elimination.

The following two sections will introduce two new couplers that – unlike three of the above mentioned couplers – do not require the solution of optimisation problems or linear systems. Both are based on the voting coupler (Section 3.5.2), but include confidence information [34].

Vote-Probweight 1 coupler

Similar to our extensions to the npSVM voting coupler described in Section 3.5.1, we did not add 1, but $r_{ij}(\mathbf{x})$ to the count for class i, $I(i)$, if i is the winning class (otherwise, r_{ji} is added to the count for class j, $I(j)$):

$$\text{if } r_{ij} \begin{cases} > 0.5 & \text{then } I(i) \leftarrow I(i) + r_{ij} \\ < 0.5 & \text{then } I(j) \leftarrow I(j) + r_{ji} \end{cases} . \tag{3.50}$$

This coupler is called *vote-probwght1*.

Vote-Probweight 2 coupler

The vote-probwght1 coupler does not reflect the fact that the probability that the data point belongs to the losing class is greater than zero. Hence, our second new coupler, (*vote-*

probwght2), adds r_{ij} to the count for class i as well as $r_{ji} = 1 - r_{ij}$ to the count for class j:

$$I(i) \leftarrow I(i) + r_{ij}, \quad I(j) \leftarrow I(j) + r_{ji}. \tag{3.51}$$

The advantage of the new couplers is that they have low implementational and run-time requirements. In contrast to the HT, Markov and minpair couplers, they do not require the solution of an optimisation problem or of a linear system. Unlike the voting coupler, which can lead to ties in the class predictions because two or more classes can have the same number of votes, the new couplers will always predict only one class. Sections 8.6.3 and 8.6.4 will discuss experiments using npSVM and pSVM couplers to see whether the newly proposed couplers are competitive. These experiments on text/spoken document data supplement the experiments performed by Wu et al. [95]. They have tested the above mentioned pSVM couplers (except, of course, the two newly introduced couplers) on many data sets, but not on text data. In fact, this thesis incorporates the first thorough investigation of conventional and probabilistic couplers on text classification.

3.6 SVMs and Structural Risk Minimisation

This section points out the relationship between SVMs and the principle of Structural Risk Minimisation (SRM), which is part of Statistical Learning Theory. It will help to explain why SVMs perform successfully for many classification applications.

One characteristic of SVMs is that they incorporate an “overfitting protection”. The error on an unknown test set will usually not be much higher than the error on the training set. The reason is that the margin between two classes is maximised; the following paragraphs will make this link clearer.

3.6.1 Statistical Learning Theory and Structural Risk Minimisation

Statistical Learning Theory, also called Vapnik-Chervonenkis Theory and introduced by Vapnik [88], deals with how to control the generalisation (or prediction) ability of a learning machine. It provides a theory to formally describe this ability, in contrast to other methods, which provide only heuristics.

The principle of Structural Risk Minimisation (SRM) [77, 29, 17] links the complexity of a learning machine to its prediction ability [17, p.45]. The complexity can informally be characterised as the variance of the decision boundary; for example, a linear boundary is less complex than a 5th degree polynomial. For decision functions linear in parameters such as polynomials, the complexity is the number of free parameters (e.g. in case of polynomials, their degree). For other functions, the VC-dimension can be used as a measure of complexity [17]. Other measures exist, for example annealed VC entropy or fat shattering dimension [77].

VC-dimension The VC-dimension measures the complexity of a class of discriminating functions, not of a single function. Consider m data points with class labels $y \in \{\pm 1\}$. There are 2^m different class combinations (permutations) for m points. A set of functions that is able to separate *all* 2^m class combinations is said to *shatter* m points. The VC dimension of a set of functions is defined as the highest number of points m that it can shatter. This does not mean that it can separate *any* m points, but that there exists at least one set of m points that can be separated. For example, in an n-dimensional space, the set of all possible linear hyperplanes has a VC-dimension of $n + 1$.

Risks Two different types of errors can occur in statistical learning: training error and test error. The former is also called *empirical risk* R_{emp}, while the latter is called *risk* R. The goal of machine learning is to find a function $f(\mathbf{x})$ that correctly classifies unseen examples $(\mathbf{x}, y)$ so that $f(\mathbf{x}) = y$. In other words, a f is to be found that minimises the risk R. However, the risk is not known; what is known is the empirical risk that can be calculated from the training data alone:

$$R_{\text{emp}}[f] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |f(\mathbf{x}_i) - y_i|. \tag{3.52}$$

One can of course use a test set and compute the classification error on it; but this is not the true risk, just an error on *one* test set out of many. The true risk is the error on all possible test data, and therefore a purely theoretical measure.

Empirical Risk Minimisation and Structural Risk Minimisation Some machine learning algorithms, like neural networks, try to chose a f that minimises the empirical risk and hope that the risk becomes minimal (Empirical Risk Minimisation, ERM). One severe danger of this approach is overfitting: the decision function adapts so well to the training data, i.e. becomes so complex, that it can hardly generalise and will produce a high test error (see Figure 3.7 (a)). Model selection techniques like cross-validation have to be applied to

avoid overfitting. On the other hand, if a simple decision function is chosen by, for example, restricting the decision function to be linear instead of a high-degree polynomial, there will be a high empirical risk. This issue is also known in classical statistics as the bias-variance dilemma [77, 29]: In order to keep the variance (complexity) low, the class of possible decision functions has to be restricted, i.e. a high bias has to be imposed. A low bias allows functions with many degrees, which then become complex. The challenge in machine learning is to find the right balance between over- and underfitting, between bias and variance. The empirical risk is a bad indicator of whether optimal balance was achieved.

Structural Risk Minimisation provides a formal approach to this dilemma. In order to estimate an upper bound of the risk, a confidence interval term $\phi(h, N, \eta)$ is introduced that links risk and empirical risk:

$$R[f] \leq R_{\text{emp}}[f] + \phi(h, N, \eta). \tag{3.53}$$

This inequality holds with a probability of $1 - \eta$. $h < N$ is the VC-dimension of the class of functions that can be implemented by the learning machine. The confidence term

$$\phi(h, N, \eta) = \sqrt{\frac{h(\log \frac{2N}{h} + 1) - \log(\eta/4)}{N}} \tag{3.54}$$

increases with increasing VC dimension $h \in \mathbb{N}$. The interaction of the elements of (3.53) is depicted in Figure 3.7 (b). Note that (3.53) does not make it possible to compute the true risk (which is, of course, unknown), but gives an upper bound of the true risk, which will probably not be crossed. It turns out that this upper bound is very conservative, the real risk is much lower than the bound [44].

A simple hypothesis space with a small VC-dimension (e.g. h_1 in Figure 3.7 (b)), will probably not contain good approximating functions and will lead to a high training error. An overly complex hypothesis space with a large VC-dimension (e.g. h_m in Figure 3.7 (b)) may lead to a small training error, but the second term in the right-hand side of (3.53) will be large, so that the test error might be high.

The SRM principle focuses on decreasing the VC dimension, while keeping the training error zero (in case of separable training patterns without outliers) or low. It has to be emphasised that SRM minimises the upper bound of the true risk, while ERM focuses on the empirical risk alone. Hence, in contrast to ERM, SRM provides overfitting protection by selecting the complexity of the decision function in such a way that a minimum upper bound on the risk is found.

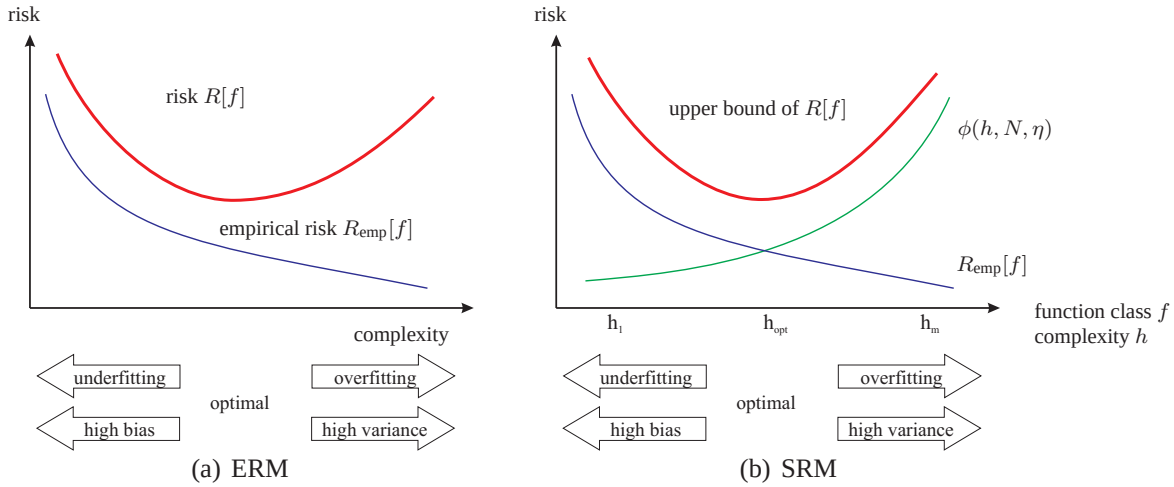


Figure 3.7: Risks in Empirical and Structural Risk Minimisation. ERM focuses on minimising the empirical risk, hereby missing the minimum risk, whereas SRM minimises an upper bound of the risk. The upper bound is the right hand side of eq. (3.53).

3.6.2 Linking SRM and SVMs

The VC dimension of the class of all linear boundaries in an n-dimensional space is $n + 1$. If one selects only a subset of these boundaries, the VC dimension decreases: Linear hyperplanes that satisfy (3.12) (canonical hyperplanes) and whose margin is at least δ_0, or $\delta = 1/\|\mathbf{w}\| \geq \delta_0$, have a VC dimension h that is bounded by

$$h \leq \min\left(\frac{R^2}{\delta_0^2}, n\right) + 1. \tag{3.55}$$

R is the radius of the smallest sphere that contains all training data points, n is the dimension of the data space. In other words, an increasing margin will decrease the VC dimension. The margin cannot be infinitely increased, because the training error would eventually also start to increase. The two terms of the upper bound of the risk (right hand side of (3.53)) have a correspondence in the objective function of the SVM problem,

$$\frac{1}{2}\mathbf{w}\mathbf{w} + C\sum_i \xi_i. \tag{3.28}$$

The minimisation of the first term, $\frac{1}{2}\mathbf{w}\mathbf{w} = \frac{1}{2}\|\mathbf{w}\|^2$, maximises the margin, minimises the VC dimension and thus minimises the confidence term $\phi(h, N, \eta)$. The sum in the second term, $\sum_i \xi_i$, is an upper bound of empirical risk R_{emp} [30, p. 327].

These considerations show that SVMs are constructed according to the principle of Structural

Risk Minimisation. Minimal risk avoids overfitting, hence SVMs have a built-in overfitting protection. What is more, the complexity of the decision boundary – in contrast to most other classification approaches – does *not* depend on the dimensionality of the input data, as can be seen in (3.55)[4]. Usually, no feature selection (using e.g. Information Gain or χ^2 criterion) or feature transformation (e.g. PCA or LDA) is needed.

The automatic choice of best model complexity makes manual tuning of many parameters superfluous. In contrast, for classifiers built according to the ERM, one has to set the complexity of the decision function a-priori. This does not mean that when using SRM, one is exempt from a-priori choosing parameters of the learning machine: The best outlier tolerance coefficient C or kernel parameters still have to be set outside the SVM. Unlike neural networks, SVMs do not need initialisation of weights or coefficients, and hence will always yield the same decision boundary when trained with the same data.

3.7 Advantages of SVMs for text classification

One property of a text classification task is that it has got a large input space (i.e. high-dimensional (but sparse) feature vectors). SVMs can handle large dimension of data efficiently. The VC-dimension of maximum-margin hyperplanes does not necessarily depend on the number of features. If the training vectors are separated by the optimal hyperplane, then the expectation of the probability of committing an error on a test example is bounded by the ratio of the expectation of the number of support vectors to the number of examples in the training set [88]:

$$E[P_r(\text{error})] \leq \frac{E[\text{number of support vectors}]}{(\text{number of training vectors})\text{-}1} \tag{3.56}$$

This bound depends neither on the dimensionality of the feature space, nor on the norm of the vector of coefficients, nor on the bound of the input vectors. Therefore, if the optimal hyperplane can be constructed from a small number of support vectors relative to the training set size, the generalisation ability will be high, even in an infinite dimension space.

Additionally, using SVMs for text classification has the advantage that statistical feature subset selection is not necessary (see [18, 43]), and SVMs were proven to be effective for text classification [42, 43].

[4] Except for cases when the maximum margin is very small, i.e. the data are hardly separable. Then, a different choice of C might be favourable in order to enlarge the margin.

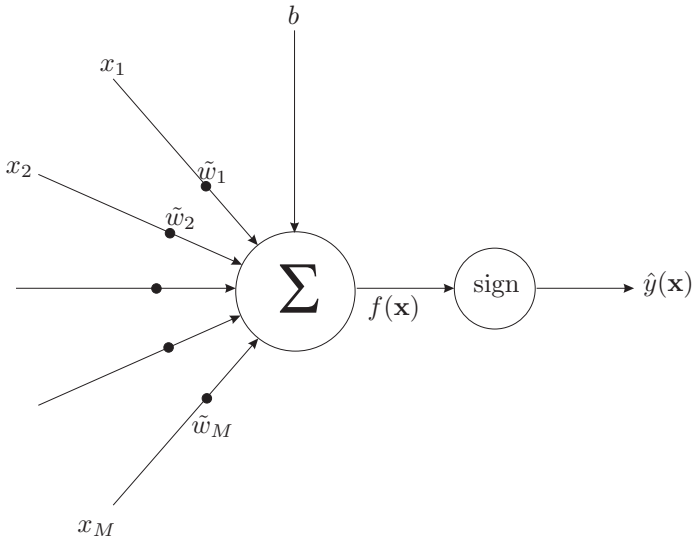


Figure 3.8: A perceptron with a sum propagation function.

3.8 Comparison to perceptrons

Support Vector Machines and perceptrons share several properties. The purpose of this section is to provide a comparison of both classifiers.

A perceptron is able to classify patterns from two linearly separable classes. The class of a pattern $\mathbf{x}$, $y(\mathbf{x}) \in \{1, -1\}$, is predicted by weighting each component of the feature vector $\mathbf{x} \in \mathbb{R}^M$ with $\tilde{w}_m$ and adding a bias $b \in \mathbb{R}$ (see Figure 3.8):

$$f(\mathbf{x}) = \sum_{m=1}^{M} \tilde{w}_m x_m + b = \tilde{\mathbf{w}} \cdot \mathbf{x} + b. \tag{3.57}$$

The sign of this function (which is usually referred to as *propagation function*) is the predicted class $\hat{y}(\mathbf{x})$ of the pattern $\mathbf{x}$:

$$\hat{y}(\mathbf{x}) = \text{sign}(f(\mathbf{x})). \tag{3.58}$$

The weights $\mathbf{x}$ and the bias b are set during the training phase of the perceptron by iteratively minimising the squared classification error of the training patterns [30].

It is therefore guaranteed that a perceptron is able to correctly classify linearly separable training patterns, since the minimum of the squared error function is 0 for such training patterns. However, the performance on previously unseen patterns can still be bad because there is no direct mechanism to control the ability of a perceptron to generalise. The perceptron's deci-

sion plane that separates the two classes lies *somewhere* between the two class clouds formed by the training patterns. In contrast, Support Vector Machines are capable to generalise well, because they minimise the upper bound of the generalisation error (called *risk* in the context of SVMs). The decision plane lies exactly in the middle of the two class clouds and separates them with maximum margin. As is shown below, the formula used for classification is similar for perceptrons (or, more generally speaking, for single-layer Neural Networks) and for SVMs. What is different is how the parameters are set during training, or to put it in other words, the objective function used for learning is different [13].

N perceptrons can be arranged parallelly to form a single layer Neural Network (where N is the number of training samples). The perceptrons' outputs are summed and a bias b is added. The sign of this sum is again used for class prediction. The bias of the individual perceptrons is fixed to 0. Classification with this network is accomplished using

$$f(\mathbf{x}) = \sum_{i=1}^{N} \tilde{\mathbf{w}}_i \mathbf{x} + b, \tag{3.59}$$

$$\hat{y}(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}(\sum_{i=1}^{N} \tilde{\mathbf{w}}_i \mathbf{x} + b). \tag{3.60}$$

Comparison of this classification rule to the classification rule of a linear SVM (3.19) yields that linear SVMs and a single layer of perceptrons predict in the same way, given that the perceptrons' weights are set to

$$\tilde{\mathbf{w}}_i = y_i \alpha_i \mathbf{x}_i. \tag{3.61}$$

There are extensions to the perceptron paradigm that make nonlinear classification possible by using the kernel trick [13]. Thus, next to the similarity between linear SVMs and perceptrons, there is also a similarity between non-linear SVMs and perceptrons.

As already pointed out, one of the differences between the two types of classifiers is the training phase. The weights of a perceptron are chosen to minimise the squared classification error of the *training* data. Moreover, *all* training patterns are used for classification (3.60). In contrast, SVMs minimise the margin of separation between the two classes, and only a subset of the training data, the support vectors, are used for classification.

As an aside it should be noted that apart from the perceptron propagation function (3.57), other types of functions exist. The sign function is often replaced by the differentiable sigmoid function. In Section 7.3.2, another type of Neural Network is described which is used for vector quantisation.

3.9 Conclusion

This chapter presented the theory of Support Vector Machines. Approaches for the classification of linearly and non-linearly separable data, for the estimation of posterior probabilities and for coupling the results of binary models to obtain multi-class predictions were treated. Four new couplers were introduced and discussed. Finally, the success of SVMs for many pattern recognition problems, especially text categorisation, has been theoretically explained.

Among the advantages of SVMs are:

- Built-in overfitting protection: since SVMs are constructed according to the principle of Structural Risk Minimisation, they aim to minimise the lowest possible test error, and thus avoid overfitting to the training data.
- The complexity of the decision boundary between two classes is independent of the dimension of the input features.
- Few parameters (usually only C and kernel parameters) have to be adjusted, which makes the learning and prediction process simple to handle.
- Using kernels, SVMs can handle non-linearly separable patterns.

These facts make SVMs good candidate classifiers for classification of texts and spoken documents.

Chapter 4

Methodology of performance evaluation

The classification techniques presented in this thesis are data driven. This means that a large training set is used to estimate the parameters of the classifiers. In addition, it is crucial to evaluate a classifier's performance with a test set. The general principle of evaluation is to compare the hypothesis of a classifier to the reference class annotation produced by humans. In theory, the reference should be perfect and unambiguous; that is why it is often referred to as "ground truth". However, annotations from different people are likely to differ. Larson et al. [45] report that two human reference annotators agreed on only 70% of the documents. Nevertheless, the existing methods for evaluation stick to the concept of a perfect reference annotation.

This chapter presents the methods used to evaluate the modules of the media monitoring demonstrator. The different types of patterns used in the demonstrator – speech signals, video signals, and automatic transcriptions – demand individual measures of performance, each of which is covered in one of the following sections.

The approach of Unsupervised Topic Discovery (UTD) presented in Chapter 10 on page 127 is inherently difficult to evaluate, since there is no fixed, pre-defined list of topics. Therefore, UTD is not evaluated by a performance measure.

4.1 Evaluation of topic classification

The following evaluation methods assume that there is only one class (topic) per document. In this thesis, multi-label settings (where more than one class is assigned to each document) will not be evaluated using a performance score.

Table 4.1: Contingency table for evaluation of classifier performance.

Hypothesis	Reference Topic is T_1	Reference Topic is not T_1
T_1	tp	fp
not T_1	fn	tn

One straightforward way to evaluate topic classifiers (and, of course, classifiers on various other domains) is to directly compare the reference class (ground truth) of each test document to the category hypothesised by the classifier. The correctness rate is the number of documents for which reference and hypothesis match, divided by the total number of test documents. This rate is used for evaluation of the HMM and Naive Bayes topic classifiers presented in Chapter 7 on page 80. It can be used for both binary and multi-class data.

More sophisticated measures allow to describe different aspects of the performance. They are presented below.

4.1.1 Measures for binary classifiers

When only two topic classes T_1 and T_2 are present, the comparison of reference and hypothesised topic of one document yields exactly one of four possible states which are listed in Table 4.1 (contingency table). The topic T_2 is referred to as *not* T_1 in this table, so that it can be used for the discussion of multi-class evaluation. The four elements, or better counts, of the table are:

- *true positive*: Both reference and hypothesis claim T_1.
- *true negative*: Both reference and hypothesis claim *not* T_1.
- *false positive*: The hypothesis mistakenly classifies the document as T_1.
- *false negative*: The hypothesis mistakenly classifies the document as *not* T_1.

For every tested document, one of these four counts is increased by one.

From the contingency table, several common performance measures can be derived for the binary class problem: precision P, recall R, miss rate M, false alarm rate FA, and error rate ER are [98, 100, 52, 48]

$$P = \frac{tp}{tp + fp} \tag{4.1}$$

$$R = \frac{tp}{tp + fn} \tag{4.2}$$

$$M = \frac{fn}{tp + fn} \tag{4.3}$$

$$FA = \frac{fp}{fp + tn} \tag{4.4}$$

$$ER = \frac{fp + fn}{tp + tn + fp + fn} \tag{4.5}$$

tp is the number of true positives in the test set; tp, fn, and tn are defined accordingly. Obviously, $M = 1 - R$. All four rates can take values between 0 and 1. If the denominator equals 0, the corresponding rate is not defined. The miss and false alarm rates can also be seen as probabilities of the classifier generating a miss or a false alarm: $P_{Miss} \equiv M, P_{FA} \equiv FA$. The false alarm rate is sometimes called *fallout*, e.g. by [98] and [52, p. 270]. While also precision and recall are likewise probabilities, the expression *probability of recall* or *precision* is not used. Instead, precision and recall are implicitly considered probabilities.

The F_1 measure is the harmonic mean of precision and recall,

$$F_1 = \frac{2PR}{P + R}. \tag{4.6}$$

A more general formulation makes it possible to adjust the relative weights between precision and recall:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}. \tag{4.7}$$

The F-measure is not a rate, therefore it is written without the percent sign (e.g. 92.4 instead of 92.4 %). Precision, recall, and the F_1 measure are widely used for evaluation of text classifiers.

4.1.2 Measures for multi-category classifiers

A multi-class problem with K (> 2) classes (i.e. more than two classes are present in the document collection) has to be broken down into several binary problems:

- Binary problem 1: T_1 — not T_1
- Binary problem 2: T_2 — not T_2
- ...
- Binary problem K: T_I — not T_I

Every binary problem is evaluated according to the scheme presented above. There are two ways to combine the binary results:

- **Macro-Averaging** treats each binary problem equally. One separate contingency table per category i is calculated, i.e. precision, recall, etc. are calculated for each category. The overall performance measure is averaged over the per-category measures. For example, the macro-averaged recall is computed as

$$R_{\text{macro}} = \frac{1}{K}\sum_{i=1}^{K}\frac{tp(i)}{tp(i)+fn(i)} = \frac{1}{K}\sum_{i=1}^{K}R(i). \tag{4.8}$$

  $tp(i)$ is the number of true positives among the test documents whose reference label is T_i . The F_1 measure, being composed of precision and recall, can be macro-averaged in two ways:

  – One F_1 score is computed for each category, which are afterwards averaged. According to [99], this is the correct way.

  – First, macro-averaged recall and precision are computed, and then F_1 is derived by taking their harmonic average.

  The first option was used to obtain the macro-averaged F_1 measures in this thesis.

  Macro-Averaging gives the same weight to the categories, independently of their size. Thus, given a very unbalanced data set, small topics may have great influence on the final performance score. The TDT workshop series [6] (see Section 7.1.1 for details) uses macro-averaging.

- **Micro-Averaging**. Only one global contingency table is kept for micro-averaging, so that the recall can be expressed as

$$R_{\text{micro}} = \frac{\sum_{i=1}^{K} tp(i)}{\sum_{i=1}^{K}(tp(i)+fn(i))}. \tag{4.9}$$

  Each class is weighted according to its number of documents in the test collection, and each document has got the same influence on the final measure. Micro-Averaging thus favours the performance on common categories. Yang [97] and Joachims [42, 43] use micro-averaging in conjunction with text classification.

As the test set used to evaluate the SVM classifier in this thesis has got very unbalanced topic sizes, as well micro-averaged as macro-averaged F_1 measures are stated in Chapter 8.

Besides the aforementioned measures, variations exist like the precision-recall breakeven point [43], or detection costs introduced by the TDT workshop evaluation [6].

4.1.3 Measures for automatically segmented documents

All performance measures presented above assume that the documents boundaries of the test set are the true reference boundaries. In other words, the topic segmentation of the test set was performed manually. Manual segmentation makes it possible to score the topic classifiers independently of the performance of the topic segmentation module.

However, a realistic evaluation of the presented automatic media monitoring system, which is the key subject of this thesis, requires that the topic classification approaches also be scored with automatic topic boundaries. These boundaries will not always be correct, so the reference and the test boundaries will not always match.

For this thesis it was decided that if a test story whose topic hypothesis is T_i has got a minimum overlap (> 0 seconds) with a reference story about the same topic, this will count as a true positive. A minimum overlap seems very small, but is nevertheless realistic. Professional media monitoring will never let an automatic system make the final decision about for which customer a story may be interesting. The system's output will always be taken as a suggestion, and manual inspection will always follow. Therefore, if the system claims that there is an interesting story at a certain time, a human will also check the stories before and after this story.

The TDT workshops take a different approach to match hypothesis and ground truth. For every time slot of the true story, the hypothesised class is recorded. The majority decision of all time slots is the final decision for the story. Thus, a hypothesised story with T_i will have to cover a great portion of a reference story about T_i in order to be counted as a match.

4.2 Evaluation of automatic speech recognisers

The word error rate (WER) (or its complement, the word accuracy WA, $WA = 1 - WER$) is the predominant performance measure of automatic speech recognisers [78]. It is defined as

$$WER = \frac{N_{\text{sub}} + N_{\text{del}} + N_{\text{ins}}}{N} = \frac{N_{\text{sub}} + N_{\text{del}} + N_{\text{ins}}}{N_{\text{corr}} + N_{\text{sub}} + N_{\text{del}}}. \tag{4.10}$$

N_{corr}, N_{sub}, N_{del}, and N_{ins} are the number of correctly recognised, exchanged, deleted (omitted) or inserted words. N_{all} is the total number of words in the reference transcription. Dynamic programming is used to match reference and recognised words.

4.3 Measures for topic segmentation

The performance of the topic segmentation module presented in Chapter 6 on page 55 is reported in terms of precision and recall. The precision rate of one news show gives the relative number of system boundaries that match the reference boundaries:

$$\text{precision} = \frac{\text{\# boundaries correctly predicted}}{\text{\# of boundaries in prediction}}. \tag{4.11}$$

The precision decreases with more incorrectly inserted boundaries. The recall rate gives the relative number of reference boundaries that are detected by the boundary classifier system.

$$\text{recall} = \frac{\text{\# boundaries correctly predicted}}{\text{\# of boundaries in reference}}. \tag{4.12}$$

The recall rate is complementary to the relative number of deleted (not detected) boundaries. A tolerance range (tolerance window) of an a-priori defined number of seconds (4 or 15) is applied when matching reference and system topic boundaries.

All results are created by counting the number of boundaries in prediction and reference for all shows (pooled), and finally calculating precision and recall from the pooled values. Another option is to calculate precision and recall individually for every show and to average the individual values. Thus, every show gets the same weight independent of its length. The performance figures of both methods usually do not differ much.

Chapter 5

Speech recognition of broadcast news

One of the key components of the media monitoring system is the automatic speech recogniser. It transforms the audio signals of a news broadcast into text; this process is called automatic speech recognition (ASR) or automatic transcription. The automatic transcription of broadcast news (BN) is one of many domains to which a speech recogniser can be applied. It is obvious that one single recogniser will not perform equally well on all domains, rather it has to be adapted to the specific domain. This chapter treats the ASR module and discusses the efforts necessary to make a speech recogniser suitable for the BN task. Two different aspects are covered: a) improving the error rate, and b) reducing memory and run-time requirements. Evaluation is done based on a 30 minutes preliminary test set, and on a 3 hours final test set.

5.1 General remarks

The predominant method to automatically recognise speech, or to be more precise, utterances like words, phones, or n-phones, are Hidden Markov Models (see Chapter 2 on page 5). It is assumed that the feature vectors of an utterance were created by a HMM. Decoding (i.e., recognition) is performed by identifying the HMM (or sequence of HMMs) that have most probably created the feature vectors (Section 2.4). Rabiner has written several popular papers on speech recognition with HMMs [66, 64].

One important tool used to improve recognition results is to consider the context a word usually appears in. This is accomplished by incorporation of an n-gram language model (LM). The LM contains many possible word n-grams and, for each individual n-gram, the corresponding probability that these n words appear in sequence. An n-gram LM also includes the lower order n-grams (for example, a trigram also contains the bigrams and the unigrams). The

probabilities are estimated using a text corpus, but since a corpus typically does not contain every n-gram that may appear in speech to be recognised, discounting or smoothing methods (see Section 7.2.1) have to be applied to give unseen n-grams a probability greater than zero. Discounting methods are not suitable for high-order n-grams because they would give too much probability mass to unseen events.

Detailed information about speech recognition with Hidden Markov Models can be found in e.g. [64, 39].

5.2 Reduction of transcription errors

The following paragraphs deal with strategies of adapting a speech recogniser for the BN domain in order to significantly reduce the number of errors in transcription. The BN recogniser evolved by constantly checking every modification and improvement to a preliminary test set (Section 5.2.1). The final test set is only applied to the best-performing recogniser (Section 5.2.2).

5.2.1 Preliminary test set

Baseline system The development of a speech recogniser for broadcast news was started with an existing recogniser trained on spontaneous speech and read sentences (mainly Verbmobil [90] data). This recogniser with its 95k dictionary incorporates a large vocabulary continuous speech recognition (LVCSR) decoder. A suitable decoding strategy especially for long-range language models (tri- or fourgrams) in combination with very large vocabularies is the stack decoder [94, 93]. It performs Viterbi search (Section 2.4.2) for the most probable hypothesis on the word level using the HMMs and the n-gram language models. The decoder sets up a stack at each time frame, where each stack contains a sorted list of word (end) hypotheses. After choosing a stack, all the stack s hypotheses get expanded simultaneously by performing a single word recognition, resulting in new hypotheses that get pushed on the specific stacks at later time frames. The decoder offers several strategies for stack selection and exclusion. Interestingly, among the several synchronous and more advanced asynchronous stack selection and inclusion strategies (time synchronous, fixed skips, conditional skips and envelope mode, see [93]), one of the simplest approaches (fixed number of skipped stacks) turned out to be the most efficient procedure for the BN transcription task.

This recogniser serves as a baseline to which modifications developed here can be compared. The dictionary, which maps the grapheme representation of a word to its phoneme representa-

tion, consists of 95k entries. The recogniser incorporates a trigram language model. Although the acoustic triphone models have got a considerable resolution of 31,780 Gaussian mixtures (Chapter 2 on page 5), the resulting transcriptions are unusable for further processing. The evaluation of this system on a preliminary test set consisting of 30 minutes of broadcast speech yield a word error rate (WER) of 79.9 % (System 1 in Table 5.1; see Section 4.2 for the definition of WER). This result suggests that the broadcast news processing task has specific requirements not met by the initial system.

Newspaper language model The first step taken to adapt the recogniser to the BN domain was to develop a news related language model. A corpus from three German newspapers, *Süddeutsche Zeitung, Frankfurter Rundschau,* and *TAZ,* covering the period from 1996 to 2000 and consisting of 400 million words, was used to create a trigram LM. Only tri- and bigrams that appear at least 3 times are included in the LM. Recognition with this newspaper LM, which reflects the statistics of written rather than spoken news language, achieved an improved WER of 72.3 % (System 2).

Monophones trained on BN The incorporation of manually transcribed broadcast news to train the phone models has a greater effect on the recognition performance. Even a monophone system trained on 50 hours of BN outperforms the general and more complex baseline system [92]: Together with the newspaper language model, the WER can be significantly reduced to 30.9 % (System 4). This system incorporates 50 different monophones and 17 non-speech acoustic models (such as pause, silence, filler, breath, cough). It is interesting to note that the monophone system with the *baseline system's LM* (instead of the newspaper LM) has a WER of 54.3 % (System 3). Thus, the newspaper LM could decrease the WER by 43 % relative for the recogniser trained on *broadcast news,* while it was only able to decrease the WER of the *baseline* system by 10 % relative.

Dictionary improvements When checking the general dictionary used for the recognisers described above, it was found out that the phonemisation was not always correct. After removing phonemisation errors (which had only effect on the recognition rate if the corrections occurred among the 5000 most probable words), words were added that appear frequently in the manual transcription of the BN training data. Different dictionary sizes between 94k and 105k were investigated, and it was found that a 98k dictionary yields the best recogniser performance [36, 35]. The improved dictionary results in a slightly better error rate of 29.1 % (System 5).

Language model interpolation As already stated above, a drawback of the newspaper language model is that it reflects written, but not spoken language. It becomes too much adapted to the type of language used in the newspapers, and cannot optimally represent the sentences spoken in broadcast news, which are not exactly structured like the written sentences. In order to overcome this drawback, a language model trained on the manual transcription of the broadcast news has been created. However, not enough BN transcriptions were available to generate a satisfactory language model. The solution is to combine the newspaper and the BN language models with linear interpolation [72]. The interpolated probability of an n-gram is calculated as

$$\begin{aligned} P_{\text{interpolated}}(w_1, \ldots, w_n) = \; & \lambda \, P_{\text{newspaper}}(w_1, \ldots, w_n) + \\ & (1 - \lambda) \, P_{\text{BN transcriptions}}(w_1, \ldots, w_n). \end{aligned}$$

The weight λ is chosen using the Expectation-Maximisation [21] (EM) algorithm.

Interestingly, although the LM size of the transcriptions is one order of magnitude smaller than the newspaper LM (12 MB vs. 162 MB in compressed format), it contributes more to the interpolated language model than the newspaper LM. ($\lambda = 0.45$). This indicates that the broadcast LM contains the relevant information in a much more concentrated way, and that the newspaper LM contains a lot of information which is of minor importance for BN transcription. The monophone recogniser with the interpolated LM is able to achieve a WER of 25.5 % (System 6).

Triphone acoustic models The introduction of context dependent acoustic models (triphone HMMs) results in a further improvement. Since the task of BN recognition requires an open dictionary with an option for periodic updates, a fixed set of context HMMs would sooner or later lead to a significantly degraded recognition performance. This problem can be avoided by a decision tree based triphone construction principle that allows a quite flexible synthesis of unknown / unseen triphones if required. The knowledge about the impact of a certain context combination to a phone can be coded in the structure of the decision tree, which in turn can be estimated on the training data and the already available monophone models. The triphone recogniser in conjunction with the interpolated LM has a WER of 19.2 % (System 7) [36]. The triphone acoustic models share 96,417 Gaussian mixture components.

Gender dependent models Further improvements could be made by additionally training the acoustic HMMs dependent of the gender. Only the means of the mixtures, and the state transition probabilities were updated [92]. The WER on the preliminary test set with these models is 18.7 % (System 8).

Table 5.1: Results of different speech recogniser systems on the preliminary test set.

System		Dictionary	Language Model	Acoustic Model Type	WER in %
1	Baseline	General	General 3-gram	Triphone	79.9
2	Baseline	General	Newspaper 3-gram	Triphone	72.3
3	BN	General	General 3-gram	Monophone	54.3
4	BN	General	Newspaper 3-gram	Monophone	30.9
5	BN	Improved	Newspaper 3-gram	Monophone	29.1
6	BN	Improved	Interpolated	Monophone	25.5
7	BN	Improved	Interpolated	Triphone	19.2
8	BN	Improved	Interpolated	Gender dep. 3-phones	18.7

5.2.2 Final test set

The final test set consists of approximately 3 hours of German TV news from the two channels ARD and ZDF. They cover the week from October 15 until 21, 2001, and thus corresponds to the final test sets used throughout this thesis (with the minor difference that elsewhere, the final test sets also cover the the preceding week).

On this test set, an overall WER of 32.7 % was observerd (see Table 5.2). This result has to be compared to the result in the last row in Table 5.1 with its 18.7 % WER. The basic difference between the two data sets, indicated by the different recognition results, is that the evaluation set comprises also a number of longer news shows. Typically, these shows contain extended interviews and reports, resulting in a higher average noise level and a significantly higher proportion of spontaneous speech. Regarding Table 5.2, it is interesting to observe that the measured WER on the ARD channel (31.6 %) equals almost the WER obtained on ZDF (32.8 %).

The results reported so far were always obtained with an audio track that has been manually pre-segmented into speech and non-speech parts. An evaluation with automatic audio segmentation of the final test set has also been performed (see last row in Table 5.2). For both channels (ARD and ZDF), the automatic segmentation causes only a loss of approx. 3 % absolute (compared to the manual segmentation). Detailed information about the audio segmentation can be found in [37].

Table 5.2: ASR Results (WER in %) on the final test set with manual and with automatic segmentation of the audio track.

	Channel		
Audio Segmentation	ARD	ZDF	Both
Manual	31.6	32.8	32.7
Automatic	34.5	35.1	35.9

5.3 Strategies for lower run-time and memory requirements

The measures described until now have an effect on the number of automatic transcription errors. To make an ASR system feasible, also the run-time and the memory requirements have to be considered. Especially for a large vocabulary system with a large language model, special methods have to be applied.

Tree organisation of lexicon Empirical studies have shown that in LVCSR systems with vocabulary sizes of 20,000 words and more, usually 90 % of the search effort has to be spend on the first two phonemes of a word. Thus, an important aspect in the context of increased efficiency in BN processing is the tree organisation of the recogniser lexicon within the single word recognition network. This tree lexicon contributes vastly to the reduction of size and the number of nodes to be expanded at each time step, because the highly redundant computation of paths for words with similar leading phones can be avoided.

LM Caching With vocabulary sizes as used for the BN recognition task, consisting of 100,000 words, the associated trigram language models grow very large. A standard trigram language model has a typical size of 150-200 MB in a compressed format(!), while a four-gram language model may even have dimensions of 250 MB and more. After uncompressing the language models, loading the acoustic models and the trigram language models, the recognition task may require 800-1000 MB RAM on a standard Linux PC, hinting that these requirements are almost doubled on 64 bit machines. A first step to reduce this need is to analyse which parts of a language model are important in terms of their frequency of access. In contrast to the general assumption that unigrams appear homogeneously distributed over time, especially in our media monitoring application the effect could be observed that words or phrases appear rather in bursts than homogeneously. The appearance of such bursts is obviously triggered by the different presented topics. This fact led to the idea of using cached disk-based language models, where only the currently relevant parts are kept in a FIFO (first

in first out) buffer and the additional parameters get reloaded, if demanded. Of course, this approach increases the disk traffic, which in turn increases the run time by a factor of 1.5-2. This increase is mainly affected by the size of the FIFO and the average story length of the processed material. A huge FIFO applied to topically homogeneous material leads obviously to a higher average number of buffer hits, which increases the efficiency of the disk-based LM. Using the cached LMs, the memory load was reduced from 800 MB for a decoding task based on a trigram LM down to 120 MB without any losses in recognition accuracy. Regarding the slow down of the recogniser, the feature of cached LMs should be considered as optional and should be used only in cases where too few memory is available. However, in these cases the cache based LM outperforms a system with a conventional memory swapping by far and guarantees a stable and predictable system behaviour.

The final speech recogniser (System 8 in Table 5.1) has got a real time factor (RTF) of about 10 on a 800 MHz PC, i.e. the automatic transcription of a 10 minute news show takes 100 minutes. This RTF suits very well the given application (media monitoring), but it would be too high for example for on-line speech recognition of a computer user.

5.4 Conclusion

The automatic transcription of broadcast news demands special efforts to build the speech recogniser. Among the numerous steps to adapt an existing recogniser to the BN transcription task, two have the greatest impact on the word error rate:

- **News language models**. The incorporation of a trigram language model based on written news (newspaper texts) reduce the word error rate from 54.3 % to 30.9 % when used in combination with monophone models trained on broadcast news. This equals to a reduction of 43 % relative. When the newspaper language model is interpolated with a language model based on spoken news (manual transcription of broadcast news), the WER drops about 12 % relative (from 29.1 % to 25.5 %). The newspaper LM has less effect when used with the baseline recogniser, which is not optimised for broadcast news.

- **Monophone and triphone acoustic models trained on broadcast news**. Broadcast news have quite a high degree of noise. Clean speech has a share of 44 %, while speech with background noise has a similarly large share of 41 % [37]. The training of monophone acoustic models based on broadcast news brings about a reduction of around one third (79.9 % $\rightarrow$ 54.3 %). The introduction of triphones instead of monophones further cuts the WER by one quarter (25.5 % $\rightarrow$ 19.2 %).

Other measures, for example improvements of the dictionary or gender-dependent acoustic models, also lead to the reduction of transcription errors, but just to a minor degree.

The representation of the dictionary as a tree, and the caching of the language model in a FIFO are crucial to make the recogniser efficient in terms of run-time and memory requirements.

The effort put into adapting an existing ASR system for the transcription of broadcast news resulted into a significant reduction of the word error rate from an initial 79.9 % to 18.7 %, or by 77 % absolute. This system is therefore promising to deliver good-quality transcriptions for the subsequent topic classification module. However, as can be seen on the final test set, its performance degrades for non-clean speech. Topic classification results with the ASR transcriptions will be presented in Section 7.3.4 on page 95, Chapter 8 on page 99 and Chapter 9 on page 122.

Chapter 6

Audio-visual topic segmentation

The topic detection module of the presented media monitoring system requires pre-segmented stories. Therefore, a news show has to be cut into topic homogeneous stories, i.e. topic boundaries have to be identified. This task is taken over by a separate module, the topic segmentation module, which will be presented in this chapter. In addition, the performance of the core algorithm of the segmentation module will be investigated with respect to a shot boundary detection problem.

Apparently, usually any detection of story boundaries requires a speech transcription beforehand. In addition, it must be stressed that the consideration of the acoustic clues does not provide enough evidence to identify story boundaries: 40 % of the story boundaries occur without a speaker turn (40 % missed), while 90 % of the speaker turns occur when no topic change is present (90 % false alarm). However, this is not the case for the audio-visual segmentation presented in this chapter. The audio-visual story segmentation module identifies the topic boundaries based on mainly the video features of a news show. One approach additionally identifies audio boundaries (e.g. speaker changes), but in no case the audio transcription is used. Two approaches to topic segmentation were investigated: a *visual*, and an *audio-visual* algorithm. Both make use of features that are derived from the video information of a news show. The audiovisual algorithm additionally uses the audio information to detect speaker boundaries. Both approaches use lattices; whereas the visual algorithm uses a lattice to represent a whole news show, the audio-visual one uses a lattice to define topic structures.

6.1 Introduction

A news lattice combines content classes and edit effects into a structure that describes a TV news show. The lattice allows only certain sequences of content classes and edit effects during recognition. A sample lattice is depicted on page 67. A lattice describes possible paths through

a news show and thus incorporates a-priori knowledge into the recognition process. The equivalent in speech recognition would be a grammar, which allows only certain combinations of words.

The design of a news lattice is an important step when building a video segmentation/indexing system. There are certain video class sequences in news shows that appear very often, but some sequences only appear once or twice. The question arises: should rare sequences be integrated into the lattice or not? Two contradicting aspects have to be considered: On the one hand, the lattice should be simple in order to limit the number of allowed paths; otherwise, rare paths might be evaluated more often than they really occur. Additionally, fewer paths allow faster decoding. On the other hand, omitting too many possible paths will not represent every detail. One might expect a worse recognition rate with more complex lattices, as they allow more paths, of which some do not represent the news show currently being processed. One goal of this paper is to find out whether this assumption holds.

Lattice complexity not only affects the recognition result, but also the time needed to build up the system: complex lattices need more time to be produced and are more difficult to verify manually. In this chapter, it will be investigated how general (simple) and complex news models affect the segmentation result.

Another important module of a pattern recognition system is the feature extraction. Features for the topic segmenter were extracted form the news video at both 12.5 frames (or features) per second (fps), and at 25 fps. The number of samples, thus the available information, doubles at 25 fps. Does this also result in better recognition rates? One important step that is common to both the visual and the audio-visual algorithm is the classification of the news show into content classes and edit effects.

The following sections explain the algorithms in more detail. Before going into detailed description, it is helpful to define the following concepts:

- A TV **station** broadcasts audio and video signals through a fixed, limited number of electro-magnetic frequencies. As one frequency (or, to be precise, one band of continuous frequencies) is referred to as a channel, the term **channel** is used synonymously for station.
- Most TV (and radio) stations transmit news broadcasts. News broadcasts that appear under the same name and at fixed times are called **programmes**. They repeat (presenting different content) usually every day, or even more often.
- A news programme consists of a series of single shows. A news **show** is a single broadcast limited in time (usually 5 to 45 minutes) and can be identified by its date, and start and end time. They do not repeat.

- A **shot** is a continuous strip of frames in a broadcast or video, separated by **edit effects** (cuts, dissolves, wipes, etc.).

One can clarify the above concepts by a relationship of sets:

$$\text{shots} \in \text{show} \in \text{programme} \in \text{channel}.$$

6.1.1 Definition of data sets

Two different data sets are evaluated in this chapter: a preliminary set and a final set.

Test set A The preliminary set of news shows (set A[1]) consists of 9 Tagesschau news shows from 1998, to amount to a total duration of 2:15 hours. They were recorded at 12.5 fps.

Test set B The final test set (set B) is made up of four different shows (Tagesschau, Tagesthemen, Heute, Heute-Journal) from two TV stations (ARD and ZDF). For this set, training of the topic segmenter was conducted on 39 different shows (15 from Tagesschau, 12 from Tagesthemen, 6 from Heute, and 6 from Heute-Journal) recorded in 2000 at 25 fps.

6.1.2 TV news indexing

The topic segmentation system was built based on the TV news indexing system by Eickeler [26, 25]. His system is able to index TV news shows from the German stations *ARD* and *ZDF*. It is limited to shows where only one newscaster appears; this means only news shows up to a length of 15 minutes (ARD) or 5 minutes (ZDF) can be classified. The indexing system classifies each image frame of a news show into content classes like newscaster, report, interview, begin, end, weather, and so on, and into edit effects like cut, dissolve, and so on. The recognition rate of Eickeler's system is 96.8 % for ARD shows (programme dependent) and 88.8 % for ZDF shows (programme independent).

6.2 Topic segmentation

Eickeler's approach was extended in two ways in order to detect topic changes in TV news:

[1]The data set abbreviations of this chapter are not related to the same identifiers used in Chapter 7.

1. After indexing a news show, rules are applied to determine the topic boundaries.
2. New lattices were defined that model, in addition to the structure of a news show, the structure of a topic inside a show.

The extraction of the features from the *visual* part of a show, however, did not change.

6.2.1 Feature extraction

Two different types of features were extracted:

- Features based on visual information only (**visual features**). Such feature vectors consist of 12 components.
- Features based on video and audio information (**audio-visual features**). These consist of the 12 visual features plus one audio component indicating speaker or audio type changes.

Visual features

The video images are stored in the YUV colour space, where the Y channel contains the luminance (grey level) of the image, and U and V are colour channels. These three values are equivalent to an image representation in the RGB space, and can be converted into RGB by a simple matrix transformation.

For extraction of the visual features, 12 numerical values are calculated for every image (frame) of the video stream. Most of the visual features rely on the difference image. Its pixels $d(x, y, t)$ state the difference of the luminance values at pixel (x, y) between two consecutive image frames at time t and time $t - 1$.

$$d(x, y, t) = |I(x, y, t) - I(x, y, t + 1)| . \tag{6.1}$$

$I(x, y, t)$ is the luminance value of the pixel (x, y) of frame t. The difference image is a good indicator for movement.

The first 7 feature components are based on the difference image. Two of them describe the centre of movement $\mathbf{m}(t) = (m_{\mathrm{x}}, m_{\mathrm{y}})^T$:

$$m_{\mathrm{x}}(t) = \frac{\sum\limits_{x,y} x \cdot d(x, y, t)}{\sum\limits_{x,y} d(x, y, t)} \qquad m_{\mathrm{y}}(t) = \frac{\sum\limits_{x,y} y \cdot d(x, y, t)}{\sum\limits_{x,y} d(x, y, t)}. \tag{6.2}$$

The variation in time of the centre of movement is also used:

$$\Delta m_{\mathrm{x}}(t) = m_{\mathrm{x}}(t) - m_{\mathrm{x}}(t-1) \qquad \Delta m_{\mathrm{y}}(t) = m_{\mathrm{y}}(t) - m_{\mathrm{y}}(t-1). \tag{6.3}$$

Two features indicate the average deviation of the centre of motion between two images:

$$\sigma_{\mathrm{x}}(t) = \frac{\sum\limits_{x,y} d(x,y,t)\,|(x - m_{\mathrm{x}}(t))|}{\sum\limits_{x,y} d(x,y,t)} \qquad \sigma_{\mathrm{y}}(t) = \frac{\sum\limits_{x,y} d(x,y,t)\,|(y - m_{\mathrm{y}}(t))|}{\sum\limits_{x,y} d(x,y,t)}. \tag{6.4}$$

A feature that is important for detecting cuts is the intensity of motion [27]

$$i(t) = \frac{\sum\limits_{x,y} d(x,y,t)}{XY} \tag{6.5}$$

with XY the number of pixels in an image. From $i(t)$, a value $i'(t)$ is computed to compensate for flashes of photographers or short-time image disruptions. It selects the smaller value of the motion intensity for frames $(t, t+1)$ and $(t-1, t+2)$:

$$i'(t) = \min\left(i(t), \frac{\sum\limits_{x,y} |d(x,y,t-1) - d(x,y,t+2)|}{XY} \right). \tag{6.6}$$

Another feature that is important for the detection of cuts is the difference histogram [27]. Its intensity is

$$h(t) = \sum_{g} |h_g(t) - h_g(t+1)|\,, \tag{6.7}$$

where $h_g(t)$ is the number of times that the grey value g appears in the image at time t. Again, a filtering similar to (6.6) reduces the effects of image disruptions:

$$h'(t) = h(t) - \mathrm{median}\big(h(t-1), h(t), h(t+1)\big). \tag{6.8}$$

The median operator removes impulsive noise, so $h'(t)$ serves as an impulse detector.

The above mentioned features are not able to detect the dissolve edit effect. This is accomplished by a special feature which is motivated by the fact that during a dissolve, the value of a pixel should be similar to the interpolated value of the neighbouring pixel values of frames $t-1$ and $t+1$. The denominator of (6.9) equals the difference of the interpolated value and

the true value at frame t. The numerator serves as a scaling factor.

$$s(t) = \sum_{x,y} \frac{d(x,y,t)}{|\frac{1}{2}(I_Y(x,y,t-1) + I_Y(x,y,t+1)) - I_Y(x,y,t)|}. \tag{6.9}$$

The last three feature components are the average values of the two colour components U and V and the average of the luminance Y.

$$m_{\mathrm{Y}}(t) = \frac{\sum_{x,y} I(x,y,t)}{XY} \qquad m_{\mathrm{U}}(t) = \frac{\sum_{x,y} I_{\mathrm{U}}(x,y,t)}{XY} \qquad m_{\mathrm{V}}(t) = \frac{\sum_{x,y} I_{\mathrm{V}}(x,y,t)}{XY}. \tag{6.10}$$

$I_{\mathrm{U}}(x,y,t)$ and $I_{\mathrm{V}}(x,y,t)$ represent the colour components U and V of the pixel (x,y) at frame t.

Audio features

Audio boundaries (e.g. speaker turns or changes from non-speech to speech) are detected using a slightly modified BIC criterion [33, 81, 87]:

The BIC algorithm takes a window of n audio features $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and arbitrarily places a boundary at position i, resulting in two segments. It then decides whether it is more likely that one single model θ_1 has produced the output $\mathbf{x}_1, \ldots, \mathbf{x}_n$, or that two different models θ_{21} and θ_{22} have generated the two segments' output $x_1 \ldots \mathbf{x}_i$ and $\mathbf{x}_{i+1} \ldots \mathbf{x}_n$ respectively. The decision rule to check if there is a boundary at point i is

$$\Delta BIC_i \stackrel{!}{<} 0 \qquad \text{with} \tag{6.11}$$

$$\Delta BIC_i = -\frac{n}{2}\log|\Sigma_w| + \frac{i}{2}\log|\Sigma_f| + \frac{n-i}{2}\log|\Sigma_s| + \frac{1}{2}\lambda(M + \frac{M(M+1)}{2})\log n. \tag{6.12}$$

Σ_w denotes the covariance matrix of all window feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, Σ_f and Σ_s are the covariance matrices of the features of the first and second segment respectively. M is the feature vector dimension. According to theory, the penalty weight λ should equal 1, but practical applications show better results with $\lambda \neq 1$.

If for a point i, $\Delta BIC_i < 0$, then also for some points j surrounding i there will be $\Delta BIC_j < 0$. The algorithm decides for the boundary to be at the point with the lowest ΔBIC value.

To detect all audio segments of a news show, the window is shifted over all feature vectors with varying lengths n and varying i. See [87] for details.

Implementing the above described algorithm, it was noticed that sometimes segment boundaries are set too early, roughly one or two syllables before the speaker finishes his or her utterance. Instead of considering the point i at which the minimum of ΔBIC occurs as a boundary, the point k was chosen that lies in the middle of the adjacent two points at which the ΔBIC value crosses the 0 line:

$$k = \frac{l+m}{2} \quad \text{with} \quad \Delta BIC_l = 0, \Delta BIC_m = 0, l < k < m. \tag{6.13}$$

This modification improves the segmentation accuracy and reduces the number of boundaries appearing too early.

As feature vectors $\mathbf{x}$ for the BIC criterion, 39-dimensional mel-cepstral feature vectors without mean subtraction were used. The penalty weight was set to $\lambda = 3.0$. The resulting boundary positions are rounded to the nearest video frame.

The audio boundaries detected by the BIC criterion are used to create an audio feature stream. The audio feature stream is extracted in such a way that the frame at which the audio boundary occurs gets a maximum predefined feature value (e.g. 1.0), whereas all other frames are initially assigned a value of 0. A predefined number of frames surrounding each peak frame (25 to each side) are assigned values that decrease linearly and symmetrically with respect to this frame. If two close audio boundaries cause an overlap of their feature values, the maximum value is taken. The result is a 1-dimensional audio feature stream that is added to the 12 video features for use in the audio-visual topic segmentation approach.

6.2.2 Modelling and recognition

Content classes and edit effects were defined that are modelled by HMMs. For test set A, the following six content classes are used: Begin, End, Newscaster, Report, Interview (an interview of the newscaster and the interviewed person) and Weather Forecast. Four classes are defined for the edit effects: Cut (a hard cut), Audio-Visual Cut (a hard video cut with an audio boundary nearby; this effect is only used by the audio-visual approach), Dissolve, Wipe and Window Change (a change of the ”topic window” next to the newscaster; this effect is used as separator between two news topics).

Content classes and edit effects

For the reference labelling of the training videos of set B, many more content classes and edit effects were used. However, several classes do not appear frequently enough to reliably train separate models. The corresponding frames were used to train another model (usually the Newscaster model). For example, the most common configuration of frames represented by the Newscaster model places the newscaster on the right hand side of the image, together with a topic window on the left side. However, to train this model, scenes are also used in which the camera has zoomed in on the newscaster, plus scenes where the positions of the newscaster and the topic window are exchanged. Some edit effects, like a circle growing (or shrinking), where the circle contains the new (or old) scene, appear very rarely and therefore could not be trained.

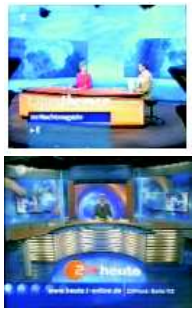
The reduction in the number of different classes to train not only makes the estimation of the models more but it also reduces the complexity of news lattices (see below).

The following Hidden Markov Models were trained on the training data of set B (most models are clarified by sample frames below them).

Content classes:

- Beginning (separately for ARD and for ZDF)

- End (separately for ARD and for ZDF)



- Intro (for ZDF only)

- Newscaster

- Two newscasters

- Newscaster1 (during interview) (only for ARD)

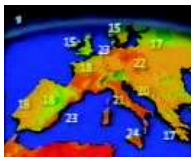
- Newscaster2 (during interview)

- Interview (only for ZDF)

- Report

- Chart/Diagram

- Weather



Edit effects:

- Cut

- AV-Cut (audio-visual cut): A visual cut with an audio cut (e.g. speaker turn) nearby. Only used for topic-structure lattices.
- Dissolve

- Wipe Left to Right

- Wipe (only for ZDF)

- Window Change. The window behind the newscaster (on the right side) contains information (pictures, text) about the topic. A change in its contents indicates the beginning of a new topic.

- Zoom-Out of Newscaster together with a Window Change (only for ZDF)

One set of models is created for every channel. The segmentation and classification of a news show is the result of calculating the sequence of HMMs that most probably has generated the observed feature vector sequence. This is done using the Viterbi decoding algorithm described in Section 2.4.2.

Lattices

A lattice is defined that combines the content classes and the edit effects of a typical and flexible news show structure. The lattices are listed in Table 6.1 on page 69, and some sample lattices are depicted at the end of this chapter. Two different approaches are taken to identify the topic boundaries:

1. **News Show Lattice + Rules; Visual Features.** The first approach makes use of a lattice that represents a whole news show and does not consider topic boundaries or the topic structure that is in a news show. This type of lattice is called *news show lattice*. The recognition result is a classification of the news show into edit effects and content classes and does not contain any information about topic boundaries. Rules are then applied to obtain the positions of the topic boundaries from the classification. For ARD, the following rules are applied:

    - All edit effects except for the Window Change effect are ignored.

- The following transitions are considered a topic boundary:
  - Begin to Newscaster
  - Report to Newscaster
  - a Window Change which separates two Newscaster scenes
  - a Window Change with a preceding Interview and a following Newscaster
  - Report to Weather Forecast
  - Everything that appears after the Weather Forecast is ignored, thus rejecting unimportant previews of other news shows.

For ZDF, these transitions are assumed to be a topic boundary:

- Begin to Newscaster, to Report or to Chart
- Zoom-Out of newscaster together with a Window Change
- Window Change to Newscaster
- Report to Newscaster
- Wipe Left to Right after a Report
- Wipe before Report, only for frames later than 3:20 minutes
- Ignore the first boundary that was detected.

The position of the first boundary of the ZDF news programmes (and also of ARD) is quite fixed, and varies only about 6 seconds. For this reason, the first boundary could be fixed to a specific time (ignoring all detected boundaries before it). However, this only makes sense if the tolerance window that compares reference and hypothesised boundaries is enlarged. See Section 4.3 for the definition of tolerance window.

Visual features (see Section 6.2.1) are used together with the news show lattice + rules approach.

2. **Topic Structure Lattice; Audio-Visual Features.** The second approach defines a lattice that is designed with the assumption that a news show is composed of several topics. Topic boundaries are embedded into the lattice. It still models a whole news show, but as it also models the structure of topics within a show. Such a lattice is called *topic structure lattice*. The beginnings of topics are marked directly in the lattice. Thus the topic boundaries are detected when decoding the videos with the Viterbi algorithm, and not as a separated step (with rules) as the News Show Lattice approach. For this approach, both visual and audio feature components are used.

Figure 6.1 on the next page depicts a simple news show lattice, while Figure 6.2 on page 68 shows a more complex variant. A topic structure lattice is depicted in Figure 6.3 on page 69.

Four different groups of lattices ($L_1 - L_4$) were defined that vary in complexity (from simple to complex), and in type (news show lattice or topic structure lattice). For every channel (L_2),

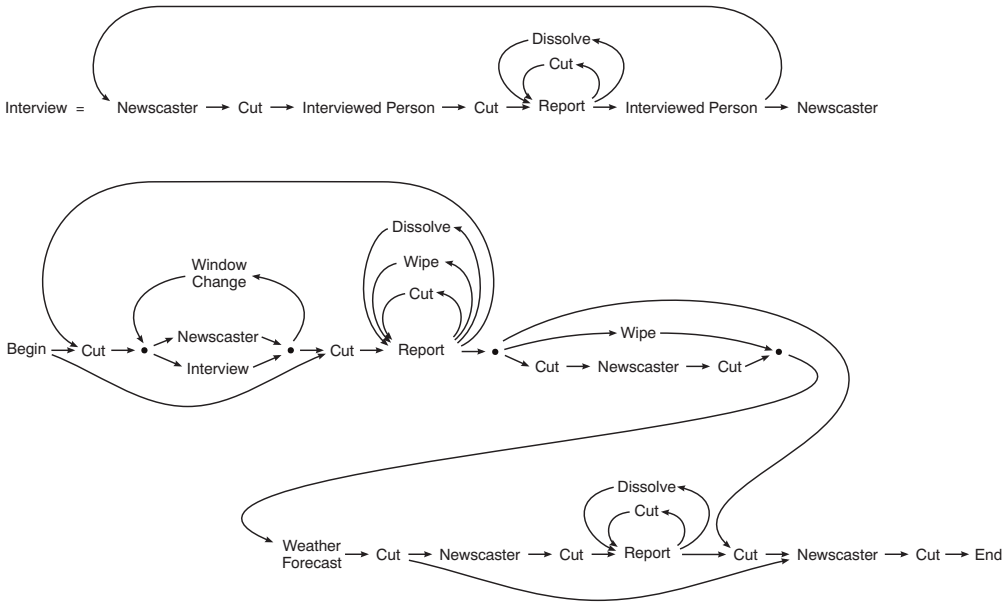


Figure 6.1: A simple news show lattice.

or even for every news programme (L_2, L_3) different lattice variants were created. L_4 was designed for and tested on ARD exclusively. The lattices were manually created based on set A or on the training set of B. Complex lattices try to capture more variations in the news shows, while simple lattices only represent the basic structures. However, even the complex lattices do not account for every possible combination, since that would make them more time-consuming to design and would allow paths that are very rare that and maybe occur only once, possibly compromising overall performance.

Table 6.1 lists the lattices together with the number of nodes and arcs that are used for their internal representation. Higher numbers indicate more complex lattices. For lattice groups consisting of more than one lattice, the minimum and maximum number of arcs and nodes is given. Lattice group L_3 has an extraordinary high number of nodes and arcs, which is only due to its internal representation and does not well reflect its true complexity: L_3 contains loops which are passed a predefined number of times. For example, the main loop in the lattice depicted in Figure 6.2 on the following page may be run 12 to 18 times. The internal representation of this lattice transforms this loop into 18 consecutive, discrete elements (of which the last 6 can be optionally skipped). Obviously, this explicit (and "brute force") representation of the loop leads to a higher number of nodes and arcs.

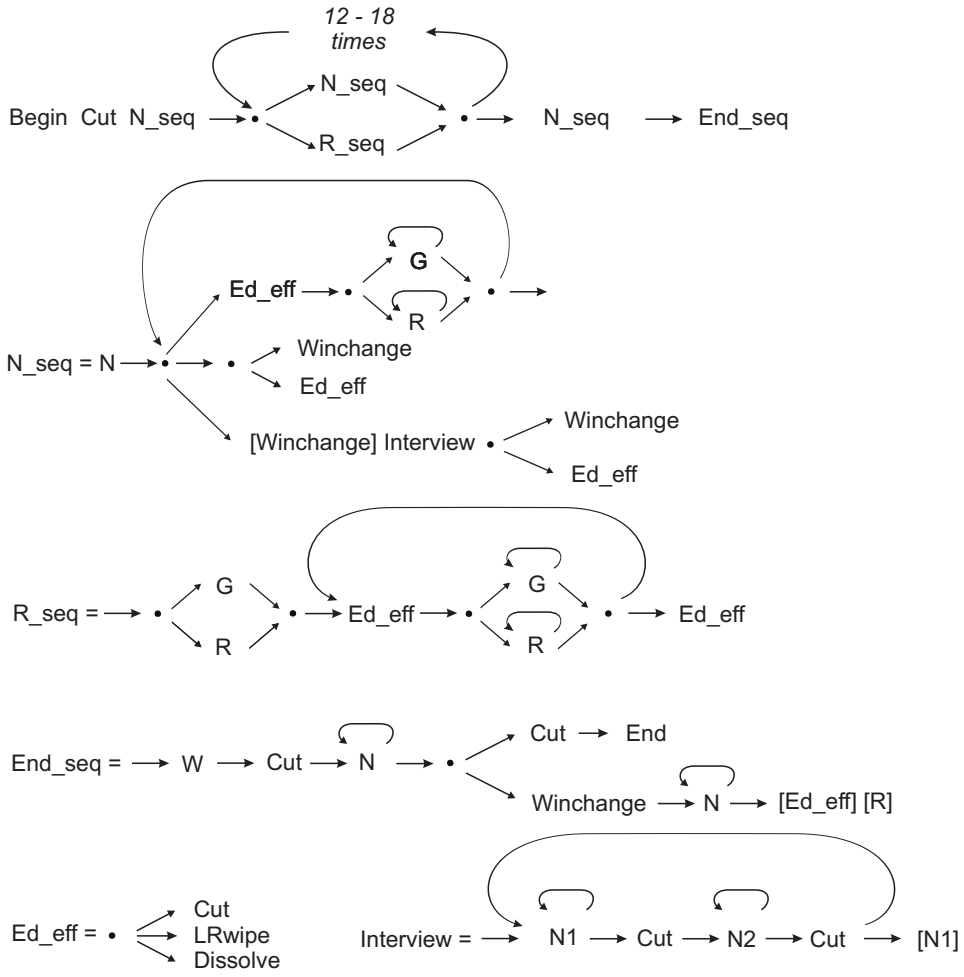


Figure 6.2: A complex news show lattice.

6.2.3 Experiments and results

Experiments were conducted on a preliminary set (data set A), and on one final set (data set B, see Section 6.1.1 for the specification of these sets). The final set consists of training and test news shows from different time periods; its test shows are also used for the evaluation of the SVM topic classifier (Chapter 3 on page 14) and of the media monitoring system (Chapter 9 on page 122). The reference boundaries were created by one person and reviewed by another person according to their personal judgement of where a topic boundary is. The experiments are evaluated using precision and recall (see Section 4.3 for the definitions). A tolerance range of 4 seconds (except where otherwise noted) was applied when matching reference and system topic boundaries.

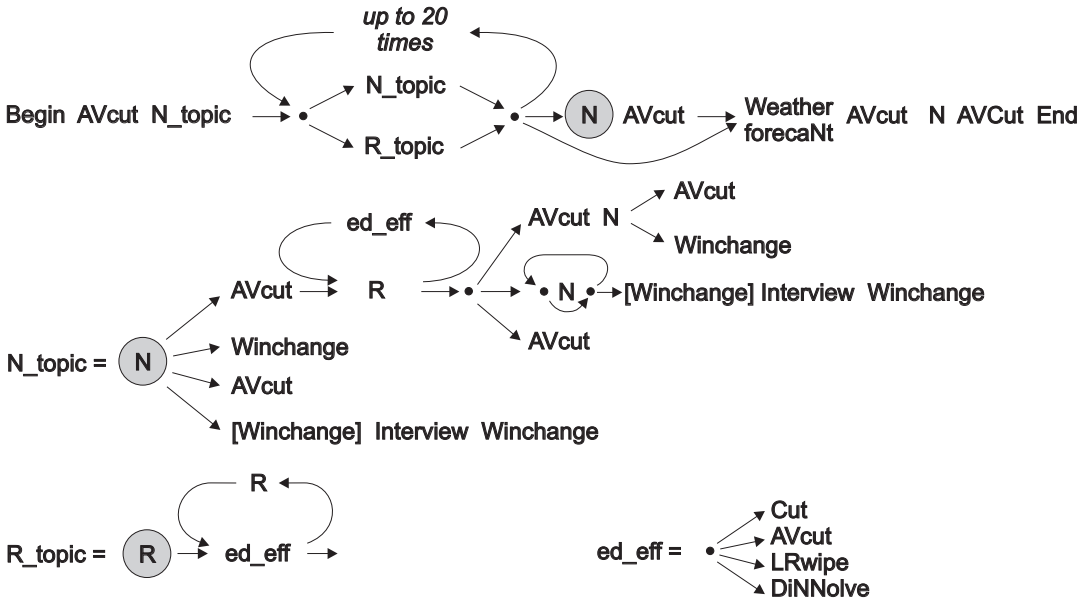


Figure 6.3: A topic structure lattice.

Table 6.1: Number of nodes and arcs in the internal representation of the news show lattices.

Lattice group	# of lattices in group	Nodes	Arcs	Designed on training set
L_1	4	30-55	75-174	B
L_2	2	59-64	147-155	A
L_3	4	793-2469	2037-6491	B
L_4	1	1121	2700	A

Test set A

For evaluation of the test set A, the hold-out method was used, i.e. each show was tested with a system trained on the other eight. As mentioned above, two different types of lattices that model the sequence of content classes and edit effects were used. The results for set A are listed in Table 6.2 [33].

Table 6.2: Topic segmentation results for set A.

Type of algorithm	News show lattice (visual features)	Topic structure lattice (audio-visual features)
Precision	88.2 %	64.8 %
Recall	82.2 %	91.5 %

Test set B

More edit effects and video content classes than in set A were defined for the final set B (see Section 6.2.2). Only visual features, but no audio features were extracted, since the audio features did not improve performance. The number of news shows per channel is listed in Table 6.3.

Table 6.3: Number of TV news shows in the final test set.

Channel	Programme	Number of shows
ARD	Tagesschau	13
ARD	Tagesthemen	2
ZDF	Heute	12
ZDF	Heute-Journal	5
	Total	32 (11 hrs 08 mins)

The following aspects were investigated by the experiments presented below:

- Feature density: 12.5 fps or 25 fps
- Lattice $L_1 - L_4$
- Different training sets for the models
- Tolerance window for matching reference and hypothesised topic boundary: 4 s or 12 s.

Feature density and lattice complexity A first series of experiments was conducted to determine whether simple or complex lattices yield better performance. These experiments use the News Show Lattice + Rules approach (Section 6.2.2). All four lattices were used for decoding, together with features extracted with both 12.5 and 25 fps. The HMMs were trained on set B. Since the reference labelling matches the training videos at 25 fps, it had to be changed to match it at 12.5 fps. Special attention has to be paid to effects that occur: for example, a dissolve of length 3 frames at 25 fps appears as a hard cut at 12.5 fps. Table 6.4 states the segmentation results on channel ARD only. Almost all experiments were also conducted on ZDF news broadcasts (not listed). ZDF results are usually 10 % to 20 % (absolute) worse than ARD results. The news shows from the ZDF channel are characterised by a greater variety of content and effects, which makes them more difficult to segment. The longer programmes of both ARD and ZDF ((Tagesthemen and Heute-Journal) have got a less rigid structure than their shorter counterparts. There are also more content classes or edit effects. For example, two newscasters never appear in the Tagesschau, but they appear in the other programmes. As Eickeler has observed [25], two newscasters pose a problem for the features used in his video indexing system, and consequently also for the topic segmentation system presented here.

Table 6.4: Topic segmentation performance (in %) with different lattices and different feature densities of channel ARD. The News Show Lattice + Rules approach is used. Models were trained on the training set of B.

Density	25 fps		12.5 fps	
Lattice	Prec.	Recall	Prec.	Recall
L_1	55.6	46.2	61.3	43.8
L_2	61.4	53.8	60.2	49.8
L_3	64.3	40.9	61.9	44.1
L_4	55.6	37.8	58.2	47.1

The Tagesschau (ARD channel) is the most important German TV news show with its market share of 35 % [20]. It is the oldest, and by far most well-known German TV news show. This coincides with its rigid, quite fixed structure, and fewer content classes and edit effects that the other news programmes. Consequently, the topic segmentation works best for the Tagesschau (see below for detailed recognition rates).

As to the question of best feature density (12.5 or 25 fps), there is no clear answer. Depending on the lattice, sometimes the one or the other result is better. However, the worst of all results are always found among the 25 fps features. The reason might be that 25 fps features are too "detailed". As one frame is usually not much different from its surrounding frames (shot boundaries are of course an exception), the doubling of the amount of features does not mean that more information is stored in the features. To put it in other words, the features extracted at 25 fps are highly correlated. At 25 fps, additional edit effects emerge that cannot be seen at 12.5 fps (e.g. the already mentioned example that a dissolve which takes 3 frames at 25 fps appears as a cut at 12.5 fps). As it is shown in Section 6.3.2, the detection of gradual transitions is more difficult than the detection of cuts. This fact could deliver another explanation why the worst results are always found among the 25 fps features.

As a consequence, 12.5 fps features should be preferred. The lower frame rate additionally has the advantage that the video material can be grabbed at the lower frame rate, yielding less amount of data (this aspect is of course irrelevant if the video was already digitalised for other purposes, because then it is likely to have the standard 25 fps).

As for the different lattices, there is likewise no clear winner. The precision rates at 12.5 fps are nearly the same, and the recall rates vary only to a slightly higher degree. Thus, it does not pay in vast the effort needed to design complex news models.

Models trained on set A Some models trained on set B contain frames which originally had different reference labels. For example, the newscaster class was trained with newscasters

at three different positions (with one being predominant). Such a "label mapping" was not performed in conjunction with set A. This is due to the fact that set B consists of much more data (11 hours of two channels vs. 2 hours of one channel), and that more programmes (two for ARD, instead of one) were used.

Although much more training data was used for set B, it might be that set A contains cleaner and more pure models, i. e. models that are required to capture less variance in the data. This is substantiated by a second set of experiments. The models that were trained on set A were used to segment the test set of set B. For 12.5 fps and lattice L_2, the precision for ARD rises from 60.2 % (models trained on set A, see Table 6.4) to 64.0 %. The recall is substantially increased from 49.8 % to 76.0%.

Table 6.5 (column "4 s Tolerance") breaks down the results on the two programmes of ARD, Tagesschau and Tagesthemen. Additionally, it contains the segmentation results on the two ZDF programmes. Again, as was already observed above, the segmenter performs worse on ZDF; it also performs worse on the longer programmes (Tagesthemen and Heute-Journal, 30 minutes each) than the shorter programmes (Tagesschau, 15 minutes, and Heute, 20 minutes).

Larger tolerance window The tolerance window of 4 s is very conservative: a predicted boundary must not appear more than 2 seconds earlier or 2 seconds later than the real boundary in order to be considered a match. The official evaluation for the TDT story segmentation task [6] uses a window size of 15 seconds. As a consequence, a larger tolerance window than 4 s was considered. It turned out that a 15 s window produces many multi-matches, i.e. a reference boundary matches more than one hypothesised boundary (or vice versa). With a 12 s window, there are significantly less multi-matches with nearly no degradation in precision and recall. The evaluation result for this window size can also be found in Table 6.5. With the larger tolerance window, the results for ARD could not be improved much (by about 3 % absolute). The results for ZDF increase by about 8 % absolute. It can be concluded that the accuracy of the boundary postitions (with respect to the reference segmentation) is much higher for ARD than for ZDF.

Topic Structure Lattice Several experiments were also conducted with 25 and 12.5 fps using the Topic Structure Lattice approach. (see Section 6.2.2). One example for 12.5 fps features and ARD data is listed in Table 6.6. It was created with lattice L_4 (this lattice was also used for the News Show Lattice approach, where its topic boundaries were ignored). It is compared to the News Show Lattice experiment that uses lattice L_2 and likewise 12.5 fps features. The figures for this latter experiment were taken from Table 6.5.

Table 6.5: Topic segmentation performance (in %) with lattice L_2, 12.5 fps, News Show Lattice + Rules approach, models trained on set A. Two window sizes of tolerance for matching true and hypothesised boundaries are investigated.

Channel	Programme	4 s Tolerance		12 s Tolerance	
		Precision	Recall	Precision	Recall
ARD	Tagesschau	68.0	77.2	71.0	79.0
ARD	Tagesthemen	45.8	68.8	50.0	72.4
ARD	Total	64.0	76.0	67.5	78.1
ZDF	Heute	50.0	55.4	58.8	64.5
ZDF	Heute-Journal	48.5	60.8	54.6	67.9
ZDF	Total	49.5	57.2	57.3	65.7

Table 6.6: Comparison of a News Show Lattice approach to a Topic Structure Approach. Results in % for ARD shows, 12.5 fps features and models trained on set A

Lattice, Algorithm	4 s Tolerance		12 s Tolerance	
	Precision	Recall	Precision	Recall
L_2, News Show Lattice + Rules	64.0	76.0	67.5	78.1
L_4, Topic Structure Lattice	49.5	70.5	53.8	75.5

Although the two experiments were conducted with different lattices, one can nevertheless draw conclusions about the performance of the News Show Lattice vs. the Topic Structure Lattice approach: As was mentioned above, the lattice has only a very limited influence on the segmentation performance. Just as for data set A, the precision with a Topic Structure lattice is significantly lower than with a News Show lattice. The recall is also lower for the current experiments, but it is higher for set B. For test set A, one can conclude the News Show Lattice approach is better than its alternative. This is also true for set B, because of the better balance between precision and recall.

Performance on set A compared to set B

The best topic segmentation for the Tagesschau shows of set B has got a precision of 71.0 % and a recall of 79.0 % (Table 6.5). For the Tagesschau shows of set A, the performance is 88.2 % precision and 82.2 % recall. Thus, recall is nearly the same for both datasets. Precision, however, decreases substantially with test set B. The reason is that test set B contains unusually long topics. Many stories are about the war in Afghanistan and the conflicts in the aftermath of the attacks on September 9th, 2001. So there are fewer topic boundaries than one would expect, and stories are longer. This corresponds with the segmentation results, because the lower precision means that there are more wrongly inserted boundaries.

One aspect when comparing the performance of the topic segmentation on test sets A and B is the time precision of the correctly hypothesised boundaries. For data set A, the size of the tolerance window is nearly irrelevant. If a topic boundary is correctly detected, there is only a difference of a few frames between the correct and the hypothesised boundary. This is different for set B. As the increase in performance from a tolerance window of 4 s to 12 s shows, the difference between correct and predicted boundary is bigger on test set B.

The decrease in performance between the two datasets can thus be accounted for by the exceptional character of test set B, not by the fact that the topic segmentation algorithms have become “worse”.

6.3 Shot boundary detection

This section does not directly deal with the segmentation of a news show into topics. Instead, it deals with the detection of boundaries between shots (not topics). A shot is a homogeneous part in a video file delimited by cuts or gradual transitions.

6.3.1 Video indexing for shot boundary detection

When a news show is classified into content classes (Newscaster, Reports, etc.) and into edit effects (Cuts, Dissolves, etc.) with Eickeler’s Video Indexing System [26, 25] (see Section 6.1.2), the shot boundaries (among other things) are detected. The performance on Cuts is very good (very few cuts are inserted or deleted), but Dissolves are not equally well classified (78 % correct detection and 44 % false detection [25]). The precision of shot boundary detection is 96 %, and the recall is 95 %; these values have been observed if the training- and the test set shows come from the same programme (programme-dependent).

It is interesting to know how well this approach works for news shows from other programmes (programme-independent identification). This issue was already investigated on a small set of four ZDF shows [26]. The TRECVID 2003 conference provides a valuable framework for further examination, since it defines and distributes a standard test set. This characteristic allows the performance of the video indexing system to be compared to other approaches.

6.3.2 TRECVID 2003

TRECVID is a series of annual workshops that emerged from the TREC conferences (see Section 7.1.1). It is devoted to the research of automatic segmentation, indexing, and content-

based retrieval of digital video. The TRECVID 2003 consists of the tasks shot boundary determination, story segmentation, high-level feature extraction, and search. The shot boundary task is to identify the shot boundaries with their location and type (cut or gradual) in the given video clip(s) [4].

The common test data for the shot boundary task consists of 5 hours of TV from the programmes ABC World News Tonight (4 shows of 30 minutes), CNN Headline News (4 shows of 30 minutes), and CSPAN (4 shows of 10 or 20 minutes, plus one show of 40 minutes). ABC and CNN broadcasts include studio scenes, indoor and outdoor scenes, while the CSPAN files are focused only on indoor scenes, such as congress debates.

Each show contains a manually created reference list of shot transitions. A transition belongs to one of the following categories: cut, dissolve, fade-out, fade-in, other. All except for the first one are gradual transitions [84]. The CSPAN shows do not contain shot transitions other than Cuts. One exception is one CSPAN show (the 40 minutes one) which contains four dissolves; these were not detected by the system. For this reason, the CSPAN news were not evaluated with respect to gradual transitions.

Evaluation

Besides defining and making available a common test set, an advantage of TRECVID is that it provides common evaluation guidelines together with a Java-based evaluation tool. The most important measures, precision and recall, are defined according to eqs. (4.11) and (4.12). When the reference and the hypothesised transitions are compared, gradual transitions can only match gradual transitions, and cuts can only match cuts. Transitions between shots are considered abrupt (a cut) if the transition's duration is 5 frames or shorter. Otherwise, a transition is considered gradual. The size of the tolerance window for matching abrupt transitions is 0.33s For gradual transitions, there is no fixed window size; one frame of the reference transition has to match at least one frame of the predicted transition. These values are much stricter than those used in the evaluation of the topic segmentation (Section 6.2.3). Precision and recall are calculated by the evaluation tool for each file, and mean precision and recall are calculated across all files.

6.3.3 Experiments

Experiments were performed with visual features (Section 6.2.1), and news show lattice L_2 designed for ARD. Models were trained on set A.

Table 6.7: Performance (in %) of Eickeler's news indexing approach on the TRECVID 2003 shot boundary test set. The numbers in bold are used in Figures 6.4 and 6.5. Empty entries: same value as in the corresponding "without" line.

		ABC		CNN		CSPAN		Mean	
LH threshold	Transition type	P	R	P	R	P	R	P	R
without	Cut+Gradual	93	71	91	61	65	86	89	67
without	Cut	93	75	90	68	65	87	88	72
with	Cut+Gradual					100	83	**93**	**67**
with	Cut					100	83	**93**	**72**
not applicable	Gradual	95	60	87	46	*no eval.*		92	53

A first series of experiments extracted the shot boundaries from the recognition result and directly compared them to the reference labelling. Results are listed in Table 6.7 ("without LH threshold"). The mean precision and recall rates (last two columns) are the output of the official TRECVID 2003 evaluation tool over all test files. The columns for each individual channel are the result of averaging precision and recall of every single show (again, the official evaluation tool was used to obtain the performance figures of the single shows). The evaluations figures were calculated separately for abrupt transitions (cuts), for gradual transitions, and commonly for both types.

The precision for ABC and CNN shows is about 90 % to 93 %, while it is only 65 % for CSPAN broadcasts. On the other hand, the recall for CSPAN is higher than for the other two programmes.

The two video files with worst evaluation results (two CSPAN shows) were investigated. For about two thirds of the wrongly inserted transitions, no apparent reason can be deduced. For the remaining insertions, three different sources of error were observed (note that the CSPAN data contains only insertions of cuts, so the following listing refers to cuts only).

1. **Fade-in and fade-out of text overlay**: Among all falsely inserted shot boundaries, fade-in and fade-out of text overlay have a share of 25 %.

2. **Movement**: Five percent of all insertions contain movement. Such a false shot boundary detection usually takes place not only because of strong movement of the main object in the scene: other factors can also result in such an insertion. For example, zoom-ins or zoom-outs also tend to generate an insertion.

3. **Transitions detected too early or too late**: Some insertions are quite close to a true transition. But because there is no frame overlap between them, no matching is possible. This situation not only generates an insertion, but also a deletion, which doubles the error rate of the recognition system. This source accounts for about 5% of errors.

One way to cope with insertions is to reject detected transitions using a confidence measure. For reasons of simplicity, only a thresholding of the log likelihood (LH) of the observed transition frames given the detected transition model was considered. Normally, one would have to compute a confidence value from the likelihood, because it is not normalised. See Section 7.2.3 for a method to calculate confidence values.

The likelihood of cuts in CSPAN videos varies less than the likelihood of the other two programmes. Inserted cuts have got a much lower likelihood than true cuts. Therefore, it is easy to find a fixed threshold for CSPAN cuts. An attempt was also made to find a threshold for ABC and CNN cuts, but use of a threshold lead to a worse performance: the decrease of insertions is far behind the increase of deletions. As far as gradual transitions are concerned, a likelihood threshold which separated insertions from true transitions could not be found.

For this reason, a second set of experiments was conducted in which a cut of a CSPAN show was rejected if its log likelihood was below -42 (see Table 6.7). As a result, almost all insertions in CSPAN vanish, and the precision rises from 65 % to 100 %. The overall precision rises from 89 % to 93 %.

The performance of the shot boundary detection is clearly better on cuts (recall of 72 %) than on gradual transitions (recall of 53 %). The precision is virtually the same regardless of the type of transition.

Comparison to other approaches

The comparison to the results submitted to the TRECVID 2003 workshop shows that the precision of both cuts and gradual transitions attained by the experimental system is among the highest, while the recall of cuts is in the middle of all performances. The recall (and precision) of gradual transitions is one of the best. Only three to four groups delivered better results. The results of the presented approach is drawn into the official result figures (see large crosses in Figure 6.4 for cuts, and in Figure 6.5 for gradual transitions). Each type of indicator represents one group; as each participating group could submit more than one run, there is usually more than one instance of each indicator type.

These are quite satisfactory results, given the fact that the video indexing system was not explicitly built for shot boundary classification. Moreover, the training data was taken from a completely different channel, and similarly, the news show lattice was not designed on the tested programmes. If a different lattice would be used, for example one with nearly no structure, but rather just content classes and edit effects in an alternating order, one might expect slightly better results.

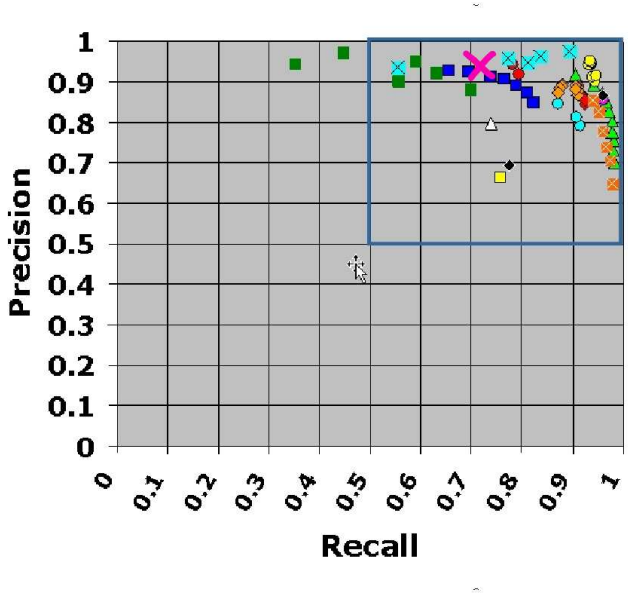


Figure 6.4: Performance in cut detection for the TRECVID 2003 shot boundary task. Results of the presented system (large cross) added to the TRECVID 2003 participants' results. Figure adapted from [84].

6.4 Conclusion

The numerous experiments presented in this chapter can be concluded as follows.

- The complexity of a TV news lattice is of minor importance. It does not need to capture every little variance of the news shows.
- A final statement cannot be made about whether it is better to extract features with 12.5, or with 25 fps. Depending on the lattice, sometimes features with 12.5 fps, sometimes features with 25 fps yield better performance. 12.5 fps should be preferred because a) the worst results were always found with 25 fps, and b) it halves the feature data size and the amount of necessary video files (compared to 25 fps).
- The News Show Lattice+Rules approach is superior to the Topic Structure Lattice approach.
- The Video Indexing system shows good performance for programme independent shot boundary segmentation. Few systems perform better on gradual transitions, and the insertion rate of cuts is also one of the best. However, the number of deleted cuts should be decreased (although several other approaches are worse), for example by introducing a simple, basic news model which does not make wrong assumptions about the structure of the tested show.

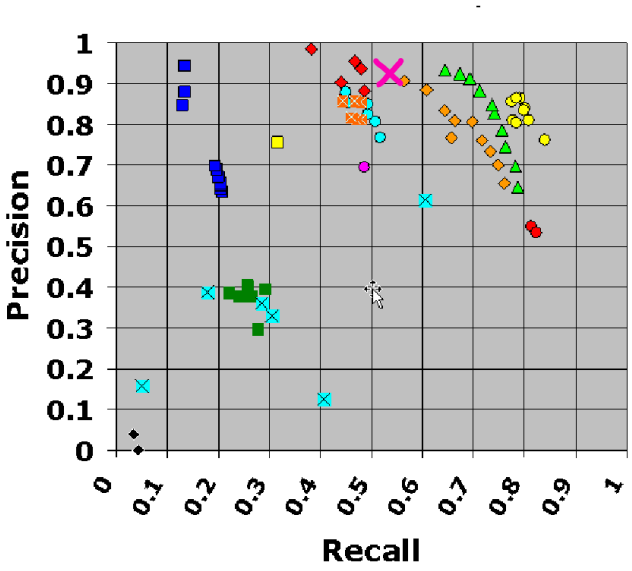


Figure 6.5: Performance in gradual transition detection for the TRECVID 2003 shot boundary task. Results of the presented system (large cross) added to the TRECVID 2003 participants' results. Figure adapted from [84].

The drawbacks of the presented topic segmentation approach are:

- Different News Show lattices have to be designed, at least for different channels.
- Likewise, the rules have also to be defined separately for every channel, even better for every programme.

The performance of the topic segmentation approach on ARD shows is very good given the fact that only the visual information of a news show (without text recognition in the video) is used. It is, however, sensitive to the programme and the channel, and degrades for ZDF news shows. The fact that the precision is lower than the recall, meaning that there are more inserted topic boundaries than deleted ones, is not too bad: the topic classification module should have less problems with over-segmented shows than with under-segmented shows having stories of more than one topic in one segment. Nevertheless, it should be promising to combine the visual topic segmentation with a topic segmentation that is based on the output of an automatic speech recogniser [85, 11].

Chapter 7

Topic classification with Hidden Markov Models

One of the key modules of an automatic media monitoring system is the topic classifier (compare Figure 1.1). It decides whether a story is relevant to a media monitoring customer, and if so, to which customer it is relevant. A story might also be relevant for more than one customer. But as this occurs very rarely, this thesis assumes that a story is relevant for only one customer.

This chapter presents and thoroughly discusses two approaches for topic classification. A third one is treated in Section 3 and Section 8. The first approach discussed here is a novel hybrid approach that combines Hidden Markov Models (HMMs) and Neural Networks (NNs). Its performance will be compared to the widely-used Naive Bayes topic classifier that serves as a baseline system. The comparison of the two will be done on the basis of data sets that contain

- As *training* data: always summaries of radio and TV broadcasts created by hand.
- As *test* data: radio or TV summaries, or automatic transcriptions of Radio news created by the ASR module presented in Section 5.

7.1 Introduction

7.1.1 The TDT workshop series

One of the most influential institutions in the field of topic classification is the TDT (Topic Detection and Tracking) workshop series [3]. It started in 1997 and has since been held every year. Unlike other conferences, which do not provide a common data set or a common set of tasks to be accomplished by the participants, both the data and the tasks are specified by the organisers. Data sets used for it are newswires, radio and television news broadcast programs,

and WWW sources. The data languages are English, Mandarin and Arabic. The following taxonomy is used by the TDT series (mostly citing [6] and [60]):

- **Event** "Something that happens at some specific time and place." Examples are: the eruption of Mount Pinatubo on June 15th, 1991, specific elections, accidents, or crimes.
- **Topic** "A seminal event or activity, along with all directly related events and activities." An event is bound to a specific time, while a topic (usually) consists of more than one event.
- **Topic Tracking** "The TDT topic tracking task is defined to be the task of associating incoming stories with topics that are known to the system. A topic is known by its association with stories that discuss it."[6]
- **Topic Detection** "According to TDT, topic detection tasks detect and track topics not previously known to the system. It is characterised by a lack of knowledge of the topic to be detected" [6]. The goal is to detect clusters of stories that discuss the topics, but not to find a topic name (label).
- **New event detection** This track used to be called First Story Detection. The goal is to find the first story in a chronologically ordered stream that discusses an *event*, not a topic. It is not necessary to name the topic (or event) associated with a story.
- **Story link detection** Determine whether two stories discuss the same topic.

The classification of topics of TV news reports, which is the subject of this thesis, would be called topic tracking in the language of TDT. One big focus of the TDT conferences is the classification of spoken documents (radio and TV news broadcasts). In contrast, the Text REtrieval Conference (TREC) deals with information retrieval on large text collections, such as newspapers. One participating group, the Cornell University, has been developing the influential SMART retrieval system [15] for 30 years The SMART system was the first one to introduce the vector space model (see Section 8.3.1).

7.2 Naive Bayes classification

This section presents a well-known classification approach known as Naive Bayes [23]. It assumes that the words w_i in a document d appear independently of other words. The likelihood of a document $d = (w_1, w_2, \ldots, w_n)$ given a topic T_j can then be expressed as

$$P(d|T_j) = P(w_1, w_2, \ldots, w_n|T_j) \underset{\text{Naive Bayes}}{=} \prod_{w_i \in d} P(w_i|T_j). \tag{7.1}$$

The topic with the highest likelihood is the classification result:

$$\hat{T} = \underset{j}{\operatorname{argmax}}\, P(d|T_j). \tag{7.2}$$

Alternatively, the topic can be chosen according to the maximum a-posteriori (MAP) principle using the Bayes formula:

$$\begin{aligned}\hat{T} = \underset{j}{\operatorname{argmax}}\, P(T_j|d) &= \underset{j}{\operatorname{argmax}}\, \frac{P(d|T_j)P(T_j)}{P(d)} \\ &= \underset{j}{\operatorname{argmax}}\, P(d|T_j)P(T_j).\end{aligned} \tag{7.3}$$

The assumption that words are independent is obviously not true, but Naive Bayes performs well in practice. With Naive Bayes, a topic can be modelled by unigrams[1]. It has the advantage of easy implementation, and is therefore often used for comparison to other classification approaches ([42], [101], [68], [45]). It it used here for the same reason. Rennie et al. have presented an improved version of Naive Bayes, which approaches the accuracy of SVM classifiers [67]. They represent the words by a normalised TF-IDF measure (compare Section 8.3.2). Classes are modelled by all documents *not* in that class, giving better estimates due to a larger training set. This tackles the bias of the decision boundaries that is introduced if classes largely vary in number of training documents. The wrong independence assumption is compensated for by normalising the weight of each (word, class) pair. Their improvements are, however, not implemented here.

7.2.1 Estimation of $P(w_i|T_j)$

The most straightforward method to estimate $P(w_i|T_j)$ is to count the relative number of words that appear in the stories about T_j, and then to divide it by the total number of words in these stories. This estimate, which is a maximum likelihood estimate [52], has an inherent problem: If a word w_i does not appear in the training stories of T_j, then $P(w_i|T_j)$ will be zero. A test document that does contain w_i will then result in a likelihood (7.1) of zero. This phenomenon is known as the zero frequency problem.

Several approaches can be used to avoid $P(w_i|T_j) = 0$. They can be classified into discounting methods (which change the words counts) and smoothing methods (which change the probabilities). Discounting methods add or subtract a small number ϵ to the count of every term

[1] A unigram is the probability of a word occurring independent of others. A bi-gram, for example, is the probability of a word appearing after another word.

(and afterwards normalise the counts to form probabilities). Smoothing methods can again be subdivided into two classes. One variant adjusts the probability of every term by interpolation with a background probability: $P_{\text{new}}(w_i|T_j) = \lambda P(w_i|T_j) + (1 - \lambda)P(w_i)$. The other variant adjusts the probability only of unseen words (backing-off).

7.2.2 Implementation of Naive Bayes

For the presented media monitoring system, the Naive Bayes approach was implemented using single-state discrete Hidden Markov Models. Each HMM models one topic T_j by estimating the topic-conditional word probabilities $P(w_i|T_j)$. They are estimated with backing-off (see Section 7.2.1) so that all unseen words get a minimum probability of $2 \cdot e^{-6}$.

During classification, the likelihood of a document is calculated for all training topics T_j with

$$P(d|T_j) = \prod_{w_i \in d} a_{11} P(w_i|T_j). \tag{7.4}$$

The appearance of the first state self-transition probability, a_{11}, is due to the representation as HMMs. This formula does not exactly equal the Naive Bayes formula (7.1), but keeps its spirit. The topic with the highest likelihood (7.4) is chosen as the classifier's prediction result.

7.2.3 Confidence

The *Off-topic* class (in other contexts also called General English [54], or background class) needs special consideration, because it contains all the stories that do not belong to any other topic. In theory, it contains an infinite number of associated stories, but there is usually no training data for it. Two approaches are used to tackle this problem:

1. Taking a large set of, or all available training data, and use it to train the **background model**. It is hoped that the large variety in all training data reflects the variety in the background class.

2. Compute a **confidence** value for each classification result that tells how sure the classifier is about its decision. This approach does not use a background model. If the confidence value is below a given threshold, the classification is rejected, and the background (here: *Off-topic*) class is assumed to be the correct class.

For the presented Naive Bayes approach, the confidence approach was used. A background model was trained together with the Support Vector Machine classifier (see Chapter 8 on page 99).

The calculation of the confidence measure was implemented as follows. The N best topic hypotheses for every story are computed (with $N = 30$). The topic predictions are ranked according to their log likelihood, $\log(P(d|T_j))$. The confidence measure for the best topic is calculated as the "distance" (i.e., the difference) of its log likelihood to the averaged log likelihoods of the other stories:

$$Conf = \frac{ll(1) - \sum_{n=2}^{N-1} ll(n)/(N-2)}{ll(1) - ll(N)}. \tag{7.5}$$

The denominator normalises the measure, which is necessary because the likelihoods (and hence their difference) are not normalised. $ll(n)$ is the log likelihood of n-th best result. This confidence measure was proposed by Hernández et al. [31]. Among its advantages are that there is no need for a background (off-topic) model, and that it can efficiently separate the correct results from false alarms caused by off-topic stories.

A confident classifier decision will result in a high likelihood for the best topic, and a substantially lower likelihood for the next best topics. The confidence value according to (7.5) will then be high. It will be low when the likelihoods are nearly equal. This enables one to set a threshold so that topics with confidence values below the threshold are rejected, and the corresponding story is assigned an *Off-topic* label. The threshold value can be a fixed value because the confidence measure (7.5) is normalised. Obviously, the threshold cannot be determined a-priori, but has to be determined from experiments.

Such an experiment can be found in Chapter 9 on page 122, where the Naive Bayes approach is compared to SVM classifiers. In Section 7.3.4, Naive Bayes is compared to the novel hybrid HMM/NN approach.

7.3 Hybrid classification system

The following chapters introduce a novel approach to topic classification that uses a combination of Hidden Markov Models (HMMs) and Neural Networks. It does not extract the features based on the words, as Naive Bayes, but on sub-word units. The features are quantised using a Neural Network that was trained by maximising the mutual information between the topics and the quantisation prototypes. The topics are modelled with HMMs.

Many experiments with different configurations and parameters will be presented. For most of them, the test set did not consist of the output of the automatic speech recogniser, but of manually written summaries.

7.3.1 Feature extraction

The text source from which feature vectors are extracted are summaries of TV news in German, but it might also be the output of the automatic speech recogniser module (Section 5). In a pre-processing step, numbers and punctuation marks are removed and all characters are converted to lower case. Then, a sliding window W of size w (typically $w = 1..6$) characters scans the text. From each character C in the window, a 32-dimensional binary feature vector $\mathbf{x}_C$ is extracted (Figure 7.1). Exactly one component of $\mathbf{x}_C$ gets a value of +1, the others are assigned a value of 0 (in unipolar case) or -1 (in bipolar case). The vector representing an 'a' has a value of +1 at its first component, a 'b' gets a +1 at the second component, and so on. The feature vector of each text window $\mathbf{x}_W$,

$$\mathbf{x}_W = [\mathbf{x}_{C_1}\ \mathbf{x}_{C_2} \ldots\ \mathbf{x}_{C_w}], \tag{7.6}$$

thus has a size of $w * 32$ with w components being +1.

These vectors are called *frames*. The resulting concatenated feature vector is made up of f adjacent overlapping frames: $\mathbf{x}_R = [\mathbf{x}_{W_1} \ldots\ \mathbf{x}_{W_f}]$. Hereby, not only the context of a window is considered, but also characters that are in the centre of the window are duplicated, which improves the recognition result (see below).

Thus, in contrast to standard approaches, where the features are based on the words (e.g. Naive Bayes in Section 7.2, or SVMs, [43]) our approach uses VQ labels to represent feature vectors generated from *character sequences* and is therefore independent of a word lexicon. The idea behind using such feature vectors is that if characters are wrongly recognised by e.g. speech recognition (or OCR for other applications), the distance between the vector of the correct spelling and the vector with one wrong character is the same whatever the wrong character may be. Scanning with a window, provided its size is properly chosen, emphasises the morphemes of the text, and thus the semantic information carriers. A morpheme is the smallest unit of language that contains information, or meaning. For example, in the word *running*, the morpheme *run(n)* indicates that the word has something to do with running, while *ing* tells something about its grammatical property.

7.3.2 Vector quantisation

To reduce the dimensionality (typically 64 ... 224) of the vectors $\mathbf{x}_R$, they are quantised using J prototype vectors. Each of the prototype vectors $\boldsymbol{\mu}_j$ gets an index number (label of the j-th partition) m_j, $1 \leq j \leq J$. Then, each vector $\mathbf{x}_R$ is represented by the number (label) of its nearest prototype vector $\mathbf{x}_R$. A good choice of prototype vectors will map morphemes

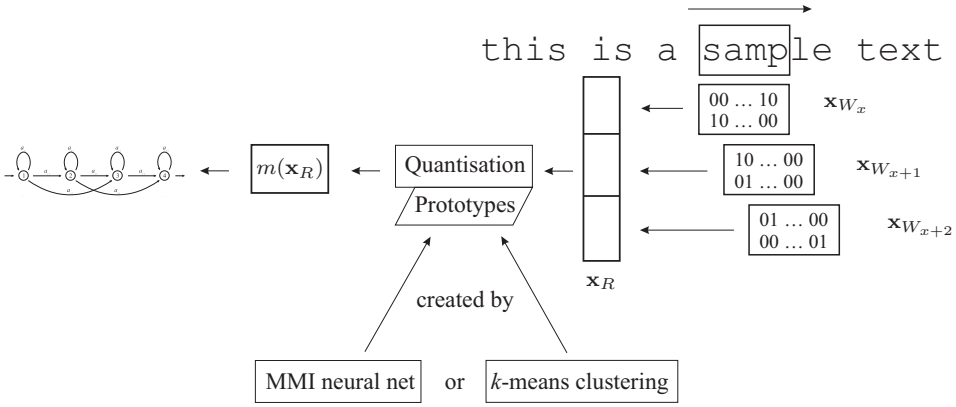


Figure 7.1: Overview of the architecture of the HMM topic classifier.

with single misrecognised characters, inserted characters or with a changed stem vowel to the correct morpheme, and thus increase the robustness of the system.

Two methods were investigated for creating the prototypes. The first one clusters the feature vectors of the training set using the k-means clustering algorithm [51].

Neural network optimisation criterion

The second method trains a single-layer neural network (no hidden layers). Its optimisation criterion is to maximise the mutual information (MI) $I(\tilde{T};M_\Theta)$ between the prototype vector labels m and the topics T. $\tilde{T}$ is the sequence of topics T of the training data, and M_Θ is the sequence of the prototype labels m that correspond to the feature vectors of the training data. The prototype label $m(\mathbf{x})$ of a feature vector $\mathbf{x}$, i.e. the label of the prototype vector that is nearest to $\mathbf{x}$, depends on the choice of the neural network parameters Θ, hence the subscript of M_Θ.

The choice of the mutual information as the optimisation criterion can be motivated in two ways:

- The vector quantisation results in an information loss. If prototypes with a high mutual information between all m and all T are used, this loss is reduced.
- Neukirchen et al. [55, 57] have shown that choosing the prototypes (in other words, the neural network parameters Θ) that yield the highest mutual information is equivalent to choosing them in such a way that they maximise the likelihood of the training feature

vectors given their respective classes (maximum likelihood criterion), and also according to the maximum a-posteriori (MAP) criterion:

$$\Theta_{ML} = \underset{\Theta}{\operatorname{argmax}} \prod_{n=1}^{N} p_{\Theta}(\mathbf{x}(n)|T(n)) \tag{7.7}$$

$$\Theta_{MAP} = \underset{\Theta}{\operatorname{argmax}} \prod_{n=1}^{N} \frac{p_{\Theta}(\mathbf{x}(n)|T(n))}{P(\mathbf{x}(n))} \tag{7.8}$$

$$\Theta_{MMI} = \underset{\Theta}{\operatorname{argmax}}\, I(\tilde{T};\ M_{\Theta}) = \Theta_{ML} = \Theta_{MAP} \tag{7.9}$$

Both the maximum likelihood and the maximum a-posteriori criterion are very popular methods to choose correct parameters. The equation that links the general ML and MAP criteria to the specific case of the vector quantiser is [70, 57]

$$p(\mathbf{x}|T) = \frac{p(\mathbf{x})}{P(T)} P_{\Theta}(T|m(\mathbf{x})). \tag{7.10}$$

The probabilities are subscripted by Θ if they depend on the choice of Θ.

Neural network layout

The prototype vectors $\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_J}$ are the weights of the neural network, i.e. its parameters θ. The optimisation criterion used to learn the VQ parameters can be formulated as

$$\Theta_{\text{VQ}} = \underset{\Theta}{\operatorname{argmax}} \{I(\tilde{T};\ M_{\Theta})\} = \underset{\Theta}{\operatorname{argmax}} \{H(\tilde{T}) - H(\tilde{T}|M_{\Theta})\}. \tag{7.11}$$

As Θ is independent of the topic labels $\tilde{T}$, (7.11) can be rewritten as

$$\Theta_{\text{VQ}} = \underset{\Theta}{\operatorname{argmin}} \{H(\tilde{T}|M_{\Theta})\}. \tag{7.12}$$

The optimal parameters $\Theta_{VQ} = \{\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \ldots, \boldsymbol{\mu_J}\} = \{\mu_{11}, \mu_{12}, \ldots, \mu_{1M}, \mu_{21}, \ldots, \mu_{JM}\}$ can be determined by a gradient descent approach [12]. μ_{jm} is the neural network weight between input node m and output node j, and at the same time the m-th component of the j-th prototype

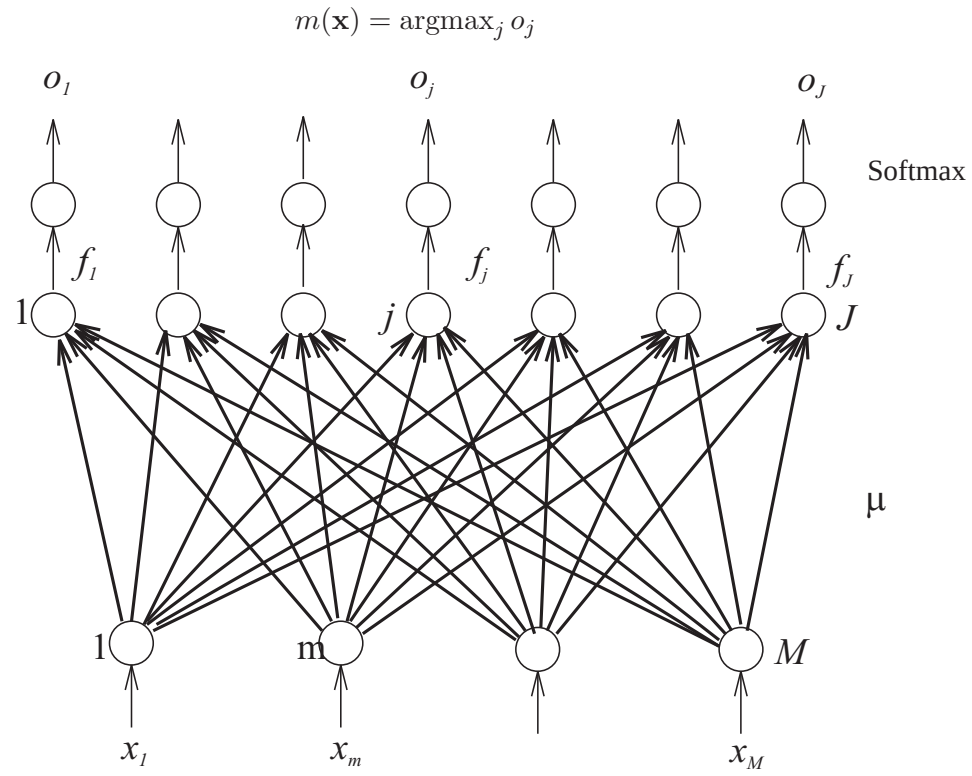


Figure 7.2: Neural network used for creation of VQ prototype vectors according to the maximum mutual information principle (Figure adapted from [56]).

.

vector $\boldsymbol{\mu}_j$. The gradient of $H(\tilde{T}|M_\Theta)$ with respect to a NN parameter μ_{jm} can be expressed as [55]

$$\frac{\partial H(\tilde{T}|M_\Theta)}{\partial \mu_{jm}} = \sum_{n=1}^{N}\sum_{i=1}^{J} 2 \cdot (x_m(n) - \mu_{jm}) \cdot \delta_{ik} \cdot A_i(\mathbf{x}(n)) \tag{7.13}$$

with

$$A_i(\mathbf{x}(n)) = \frac{C}{N} o_i(\mathbf{x}(n)) \cdot \left(\log P(T(n)|m_i) - \sum_{k=1}^{J} \log P(T(n)|m_k) \cdot o_k(\mathbf{x}(n)) \right). \tag{7.14}$$

$\mathbf{x}(n)$, $1 \leq n \leq N$, is the n-th out of N training samples. $x_m(n)$ is the m-th component of the M-dimensional vector $\mathbf{x}(n)$. The output activation function $f_i(\mathbf{x})$, which describes the

output of neuron i when the feature vector $\mathbf{x}$ is presented to the input layer, is the (negative and squared) Euclidean distance between $\mathbf{x}$ and the i-th prototype, $\boldsymbol{\mu}_i$:

$$f_i(\mathbf{x}) = -\sum_{m=1}^{M}(x_m - \mu_{im})^2. \tag{7.15}$$

This function is motivated by the fact that the features will be quantised to their nearest prototype vector. The negative sign in (7.15) has the effect that the neuron whose prototype is nearest to the input vector gets the highest activation. Equation (7.13) is only valid for a single-layer NN with a Euclidean activation function according to (7.15). The choice of the VQ label that corresponds to $\mathbf{x}$ (i.e., the result of the quantisation of $\mathbf{x}$) could be made by applying the *winner-takes-all* rule to the internal neuron activations: $m(\mathbf{x}) = m_j$, with $j = \mathrm{argmax}_j\, f_j(\mathbf{x})$. Unfortunately, this rule is not differentiable and thus cannot be used to determine the network parameters using the gradient descent approach. Therefore, the neurons' outputs $f_i(\mathbf{x})$ are non-linearly transformed by a softmax function to yield the output nodes $o_i(\mathbf{x})$ of the network:

$$o_i(\mathbf{x}) = \frac{e^{C\cdot f_i(\mathbf{x})}}{\sum_{j=1}^{J} e^{C\cdot f_j(\mathbf{x})}}. \tag{7.16}$$

The biggest double floating point number (8 bytes) that can be stored in a 32-bit Personal Computer is $2.1 \cdot e^{709}$. In order to take advantage of the whole range of double values without overflow, the constant C is set to

$$C = \frac{500}{\max_i \left\| \mathbf{x}_i - \boldsymbol{\mu}^*_{m(\mathbf{x_i})} \right\|}, \tag{7.17}$$

with $\boldsymbol{\mu}^*_m$ being the initial values of the prototypes.

Network parameter training

In order to calculate the correct weights (prototypes) of the network, the gradient of the function to be optimised, eq. (7.13), is used. In every iteration step, the weights μ_{km} are updated in such a way that the mutual information $I(\tilde{T};\ M_\theta)$ grows at a maximum rate (steepest ascend of $I(\tilde{T};\ M_\theta)$). Neukirchen [55] has shown that the Resilent Backpropagation method [69] is a good optimisation algorithm that is able to quickly find the optimal value in a few iterations. In addition, it is quite insensitive to the setting of its internal parameters (initial step width and step width update coefficient).

Before the first iteration, the weights of the neural network have to be initialised. A good

choice is to use the prototypes created by the k-means clustering, as these usually represent a reasonably quantised space and are far better than random initialisation.

After a pre-defined number of iterations (usually 10), the neural network training is finished and the resulting network parameters are used as prototype vectors to quantise any arbitrary test feature vector $\mathbf{x}_R$.

7.3.3 Topic modelling with HMMs

Each topic is modelled with a discrete HMM and using the indices of the prototype vectors, $m(\mathbf{x}_R)$, as observations. The HMM parameters are initialised by a Viterbi training and then refined using Forward-Backward training. The classification of unknown texts uses the Viterbi algorithm. A varying number of HMM states, and variation of many other parameters were examined. Details together with experiments are presented in the following sections.

7.3.4 Experiments

Test and training set

As a text source for both test and training, written summaries of TV and Radio broadcasts in German are used. Every summary was created by a professional and summarises the contents of a topic-homogeneous report.

A topic was assigned to every summary. The topic of a summary identifies the customer who wants that specific media monitoring alert. A summary is only produced from the broadcasts if it is of interest to a media monitoring customer, thus there are no *Off-topic* summaries (that could be used to train data that is not relevant to any customer). A topic consists of one or a few words and represent, for example, names of people or companies, or events. There are in the average 562 words per summary.

The following test and training sets were created:

- **A**: no stop word removal, no stemming, no text optimisation. **A1**: 22 topics in 3037 texts for training and 1319 texts for testing.
- **A2**: 173 topics in 6039 texts for training and 2700 texts for testing.
- **B**: deletion of 150 stop words, text optimisation (see following paragraph). 22 topics in 2956 texts for training and 1284 texts for testing.

In the summaries of test set A, some words are separated into two single words because they were entered in two different lines. Additionally, there are some special abbreviations. Thus, the text basis is not optimal, but the text was not changed in order to simulate in a rough way errors which are made by automatic speech recognition. For test set B, which consists of different texts, these errors were compensated for (called *text optimisation* in the list above).

There is no extra model for out-of-topic summaries, since all tests topics were restricted to the trained topics. All results were obtained with 10 iterations of both the k-means algorithm and the training of the MMI network (except where otherwise noted).

The recognition rates for the following experiments are stated as the ratio of the number of correctly classified summaries to the total number of tested summaries

Setting w and f

Initially, the best settings for the parameters w (number of characters that create one frame) and f (number of overlapping frames) have to be found. Several combinations of $w \in \{1, \ldots, 6\}$ and $f \in \{1, 3, 5, 7\}$ were examined. It turned out that $f = w = 3$ performed best. For this setting, classification results for the data sets mentioned above are listed in Table 7.1.

Table 7.1: Recognition results in % on the test sets A1,A2 and B. $w = f = 3, J = 200$, 5 HMM states for A1 and A2 and 10 states for B.

k-means	MMI	test set
48.4	47.9	A1
32.1	31.3	A2
66.0	66.6	B

k-means clustering vs. MMI neural network

The application of the MMI (maximum mutual information) neural network to produce the prototype vectors should reduce the information loss created by the quantisation. However, in almost all experiments, the classification rate did not change much (around $\pm 1\%$ absolute) compared to the k-means approach. One exception is when keywords were deleted from the test and training set (see below): the recognition rate improved from 41.3 % (k-means) to 46.7 % (MMI network).

In the following sections, the behaviour of the MMI network will be investigated in more detail.

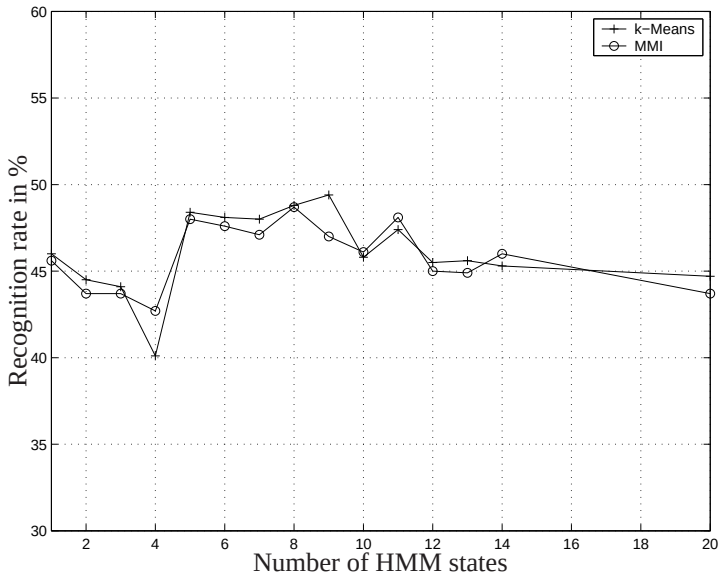


Figure 7.3: Recognition result as a function of the number of states of the HMMs. $f = w = 3, B = 200$, set A1.

Varying the number of HMM states

The number of states of the HMMs was varied from 1 to 20. Figure 7.3 depicts the results for set A1. Both the k-means and the MMI approach yield a minimum recognition rate at 4 states. They peak at 8/9 states. For test set B, a minimum recognition rate was also observed at 4 states. The best number of states for this test set is at 10 states for both the k-means and the MMI system (not depicted here). If best recognition performance is the goal, 10 states should be used for the topic identification system. However, if time is very important, one can use 5 states with only a slight decrease in recognition rate.

Changing the keywords

If the word(s) of the topic label (keywords) appear in the texts, a word search using stemming might be enough to get good recognition results. As the topic label word(s) might therefore be important to indicate the topic, they are called keywords in this sub-section. However, the keywords are not always present in the text, and even more, a speech recogniser or an OCR system might not correctly recognise all keywords. Hence, a robust topic classification system has to cope with the fact that important words do not necessarily appear in the text, but need to be induced.

To measure how well the new approach can cope with missing keywords, they were removed from the test and from the training set. In one case, the keywords were removed only from the

summaries used for testing, while in the second case they were deleted in both the test and in the training set.

Table 7.2 shows that the presence of keywords is quite important. The recognition results went down without keywords. Remarkably, the MMI network could almost completely compensate for their removal from both the training and the test set in 1-best recognition case.

Table 7.2: Results with omission of keywords. $w = 3, f = 3, B = 200$, noisy features, set A1.

k-means	MMI	keyword deletion
48.4	47.9	none
38.9	37.5	test set only
41.3	46.7	test and training set

In another experiment, a space was inserted into the middle of keywords longer than 7 characters. The idea behind this is that compound words in German consist of (sub) words that in other languages are generally written as two or more different words. As a speaker can combine words to compounds in a nearly unlimited way, a speech recogniser will tend to decode the sub-words only, and will emit them as separate words. The insertion of spaces is intended to model this effect. The recognition rate went down by around 7 % absolute when separating the keywords (see Table 7.3).

Table 7.3: Recognition results for separation of keyword compounds. $w = 3, f = 3, B = 200$, set A1, noisy features in keyword separation experiment only.

k-means	MMI	keyword separation
66.0	66.6	none
59.1	58.0	test and training set

Duplicating characters

When using feature vectors that are made up of more than one frame $\vec{o}_W$, the characters in the centre of the window are duplicated. Consider for example a window size of $w = 3$, a frame number of $f = 3$ and a text window [abcde]. The feature vector then represents the characters [abcbcdcde], or once a and e, twice b and d and three times c.

In Table 7.4, one can compare the results for $f = 3$ (duplicated characters) and $f = 1$ (single characters). In both cases, the same 5 characters are covered. The repetition of the centre characters leads to an improved recognition result.

Table 7.4: Effects of duplicating centre characters on set A1 and A2. $J = 200, k$-means quantiser.

Data Set	$f = 3, w = 3$	$f = 5, w = 1$
A1	48.4	45.5
A2	32.1	30.4

Noisy features

All equal text windows appearing several times in the text will lead to more than one identical feature vector. It turned out that it is better to add noise to the vectors in order to avoid singularities in the feature distribution. However, the signal-to-noise ratio should be chosen in such a way that the clusters in the feature space do not overlap.

Uniformly distributed noises with different amplitudes were investigated. Noise in the range of $[-0.005 \ldots 0.005]$ added to each component showed the best results (see Table 7.5).

Table 7.5: Effects of adding noise to the features. $f = w = 3, B = 200$, set A1.

k-means	MMI	Noise
48.4	47.9	none
50.3	49.5	$[-0.005 \ldots 0.005]$

Using more prototype vectors

As the prototype vectors have to represent the information that is in the texts in the best possible way, the right choice of the number of prototypes is important. Experiments were made with several number of prototypes on test set A1 whose results are listed in Table 7.6.

The k-means system shows its best recognition results for 500 prototypes and decreases significantly with more prototypes. The MMI system's peak is at 1000 prototypes. This indicates that the number of important lexical morphemes in our training set is somewhere in the range between 500 and 1000. The decrease in performance with a high number of prototypes might be due to over-fitting to the training set and the loss of the ability to generalise. It is expected that more prototypes will be useful if the number of topics will be increased in the future.

Observations on mutual information

The entropy of the topics in the training set of data set A1 equals $H(\tilde{T}) = 4.26$ bit (for $w = 1, f = 3$). During MMI estimation of the prototypes, the conditional entropy $H(\tilde{T}|M_\Theta)$

Table 7.6: The influence of the number of quantisation prototypes. $w = f = 3$, 5 HMM states, set A1.

k-means	MMI	# of prototypes
48.4	47.9	200
50.1	49.7	500
46.9	50.4	1000
46.7	47.3	2000

takes values of around 4.20 bit. The conditional entropy tells how much extra information on average is needed to communicate the topic label of a feature vector[2] given the quantised feature vector (i.e. the prototype nearest to the feature vector) [52]. In other words, it measures the information one gets from observing $\tilde{T}$ when M_Θ is known.

The difference between the entropy and the conditional entropy is the mutual information:

$$I(\tilde{T};M_\Theta) = H(\tilde{T}) - H(\tilde{T}|M_\Theta), \tag{7.18}$$

for the given example

$$= 4.26 \text{ bit} - 4.20 \text{ bit} = 0.06 \text{ bit}. \tag{7.19}$$

These figures show that the conditional entropy is very high compared to the entropy of the topics, and that most information gets lost when creating the prototypes and quantising them. The quantised feature vector and its topic are hardly related. This makes sense, as it is very difficult to predict the topic of a whole story from one single feature vector only. If this prediction was easily possible, there would not be a need for complex classifiers such as HMMs.

The MMI principle considers individual feature vectors, but does not take into account that only their combination, not one single feature, is characteristic of a topic. This leads to the question whether it makes sense to put effort into increasing the mutual information of single feature vectors. As one feature vector hardly allows to predict a topic, the MI will always be low; this is a consequence of the way of extracting the features. A different way of feature extraction might be reasonable, which was realised in conjunction with the SVM classifier (see Section 3 and Section 8).

Comparison to Naive Bayes

The performance of the hybrid HMM system was compared to the Naive Bayes system (Section 7.2 on page 81). The data sets used for comparison do not contain *Off-topic* texts. Con-

[2]The topic label of a feature vector is the topic of the story it belongs to.

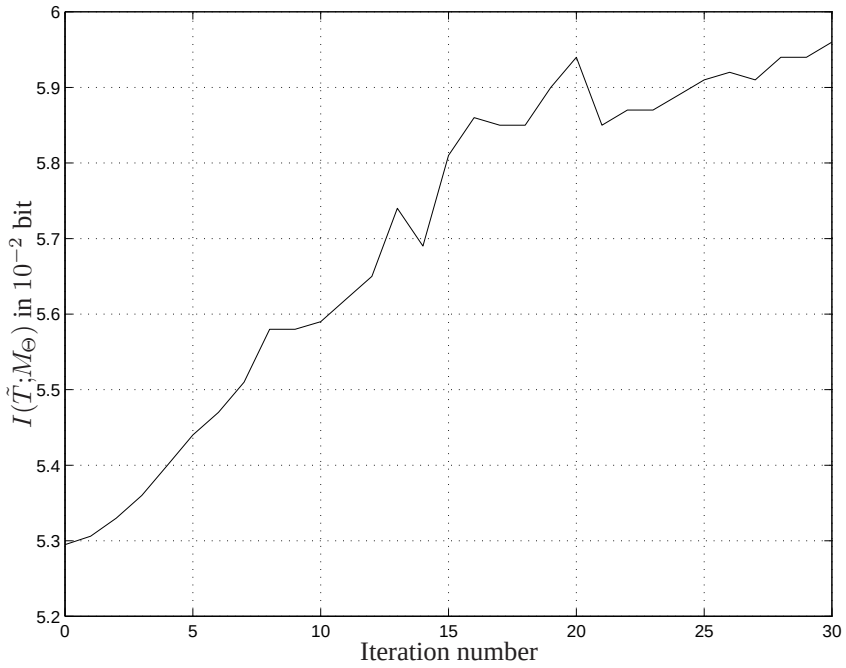


Figure 7.4: Mutual information $I(\tilde{T};M_\Theta)$ during MMI network training. $w = 1, f = 3$, 6 HMM States, $J = 200$.

sequently, the Naive Bayes approach was employed without confidence calculation and topic rejection (see Section 7.2.3 on page 83). The training data used consists of manually created summaries. Two different test data sets were used: the first set consists of summaries (the same source that was used for the training data), while the second test set consists of automatic transcriptions of radio news created by the ASR module (see Section 5). The topic boundaries for all the data used here were manually created, i.e. are "true" boundaries.

The results of the novel topic classifier and of the Naive Bayes baseline approach (see Section 7.2) for the tested summaries are listed in Table 7.7. The Naive Bayes approach works significantly better on test set B (optimal text), whereas it is only slightly better than the novel approach on test sets A1 and A2 (non-optimal text).

Table 7.7: Comparison of the best classification rates of the novel hybrid system to the Naive Bayes classifier on a test set consisting of summaries.

k-means	MMI	Naive Bayes	Test set
50.3	49.5	50.4	A1
32.1	31.3	35.3	A2
66.0	66.6	78.0	B

For tests on the ASR output, 48 reports were selected from radio news that were all on-topic,

i.e. this test set contains only reports in which a media monitoring customer is interested. These 48 reports cover 10 topics from a period of 6 days. Two different training sets were created. Both consist of summaries, but differ in number of topics. The period of the training data covers 4.5 months and ends one day before the test period starts. The first training set covers 897 topics in 113,915 summaries, but only 6 of these appear in the test set (due to the fact that the topics rapidly change with time). The second training set is limited to those 6 test topics. Consequently, the test set was also limited to these 6 topics which are covered by 42 test stories (see Table 7.8). It is common practice for the evaluation of text classification systems to choose the training and the test set in such a way that they contain only the same topics.

For the latter data set, Naive Bayes correctly classifies 67 % of the stories, whereas the novel approach is better and correctly classifies 76 % (see also Table 7.8). The Naive Bayes performance seems to be disappointing, but it has to be kept in mind that the training and the test set come from two different types of texts (hand-created summaries for training, deficient ASR transcriptions for testing). This is different from evaluations of Naive Bayes by others: there, the training and the test set come from the same text type.

The rank of performance changes when the realistic data set with its 897 training topics is used (with 10 test topics, of which 4 can never be detected): Naive Bayes has a correct classification rate of 42 %, whereas the novel approach is significantly worse with 21 % classification rate. These results show that the novel hybrid system can perform well with a limited number of training topics, but it is bad when the number of topics rises. It is not very good at choosing the correct topic from many topics, which consists a key property for media monitoring systems. But it can cope better with error-prone ASR output for a limited number of topics.

It must be pointed out that the evaluation of the novel system was performed only with a prototype codebook created by k-means clustering, not by a MMI neural network. The reason is that MMI training for 100,000 stories is not feasible: or the training features are too big to fit into memory, or (if the features are read in sequentially), the training takes too much time (several days for only 10,000 stories on a 700 MHz PC). The results with prototypes created by an MMI network would not differ much from the results from k-means prototypes, as was observed earlier in this Chapter.

7.4 Conclusion

This chapter has presented two algorithms for topic classification based on HMMs. The first one is the well-known, word-based Naive Bayes approach. It is used by many people as a

Table 7.8: Number of correctly classified stories by the novel hybrid system and by the Naive Bayes classifier on ASR test set.

Training Set		Test Set			
# Topics	# Stories	# Topics	# Stories	Novel System	Naive Bayes
897	113,915	10	48	21 %	42 %
6	4,048	6	42	76 %	67 %

baseline, or reference system. The second algorithm is a novel, character-based HMM classifier. Its feature vectors are quantised into prototypes that are created with a neural Network according to the Maximum Mutual Information (MMI) principle.

Many different experiments with different parameters of the novel system were conducted. It could not outperform the Naive Bayes approach when tested on human-created summaries (nearly error-free). But it was better at classification of ASR transcriptions (spoken documents) when it had a limited choice from only few training topics. For a large number of topics, its performance was worse. It can be concluded that it is suitable as a topic classifier for a media monitoring system if only a rough discrimination between topics (e.g. politics, economy, sports, ...) is needed. For such a scenario, it performs much better than Naive Bayes.

Chapter 8

Topic classification with Support Vector Machines

This chapter presents the experiments and results of topic classification of German and English news documents obtained with the Support Vector Machine classifier (see Section 3). It incorporates the first thorough investigation of conventional and probabilistic couplers on text classification. First, the German training and test data are described. Then, the text preprocessing and feature extraction steps are explained. A detailed discussion of the performed experiments follows that also treats experiments on the widely-used English corpus (Reuters newswire data).

8.1 Data sets

8.1.1 Test data

The German test data consists of the automatic transcription of German TV news shows, manually segmented into reports[1] (unless otherwise noted). It was taken from selected TV news, broadcast by the two stations *ARD* and *ZDF* and covering the period from 8th until 21st October, 2001. One characteristic of the test data is that a lot of reports are about the conflict and war in Afghanistan. Reports are therefore often longer than they would be in broadcasts until one month earlier.

Around 90% of the reports were assigned an *Off-topic* label, since they are not relevant to any media monitoring customer. 25 topics appear in the test set, of which two do not appear in the training set(s) used (see below). These two topics, with one test story each, were re-labelled

[1]The terms *report* and *story* are used synonymously.

to *Off-topic*. Thus, there are 23 different topics in the test set, of which only 8 occur in more than one report (including *Off-topic*). The topic labelling was adopted from the corresponding summaries of the same period. The automatic transcription was created by the ASR system described in Chapter 5 on page 47.

8.1.2 Training data

The German training data are a subset of manually created summaries of TV and radio broadcasts. The summaries were written, and topics were assigned to them by professional media monitoring employees. The data covers the period of four weeks from 10th September until 7th October, 2001. Words were not stemmed, and no stop words were removed, but numbers were deleted.

Three different training sets were created: set02(a), set03 and set04. Set03 covers those 7 topics (excluding OFF_TOPIC) that appear more than once in the test set. All summaries belonging to one of these 7 topics were taken from the above mentioned complete summary set and used to form Set03. Additionally, all those summaries which are *not* about these 7 topics were used to form the OFF_TOPIC training data.

In a similar way, set02 was created to cover all 23 topics appearing in the test set. As one topic had only two training stories, splitting it into five parts would mean that not all parts contain all topics. Therefore, it is not possible to perform five-fold cross-validation with set02. Hence, a new set set02a was produced by removing the offending topic.

The restriction of the training topics to match the test topics is motivated by the fact that in many publications, classification systems are analysed in the same way. However, one does not a-priori know which topics will appear in the news broadcasts to be monitored. A more realistic set is set04 which contains all topics which have two or more summaries. This amounts to 827 topics in 21376 summaries. One minor change was made to the *test* set when used together with this training set: It turned out that one report contains only three words. This report was removed when used together with set04.

Note that the terms set02(a), set03 and set04 describe only the training data set. The test set is equal (except the short test story that was removed when used in conjunction with set04.

Table 8.1: German data sets for training and testing the SVM classifier.

Set	Classes in training set	Training docs	Test docs
set02a	22	4825	443
set03	7	3065	443
set04	827	21376	442

8.2 Text preprocessing

Stop word removal and stemming may be used under the assumptions that stop words are irrelevant for the classification task and that different words based on the same stem are equivalent with respect to the classification task. As for the effectiveness of stop word removal, there are contradicting results in the topic classification literature [50]. Lo and Gauvain [50] report that stemming, but not stop word removal improves their results for the TDT topic tracking task (cf. Section 7.1.1). Leopold and Kindermann [46] have observed that for text classification with SVMs, stop word deletion is not necessary. Joachims [43] does not use stemming or stop word removal with most of his SVM experiments. So the question arises whether stop words should be removed and words should be stemmed.

Another aspect is that the test documents are the output of an automatic speech recognition (ASR) system. If a word is not included in the ASR vocabulary, it will not be recognised or it will be recognised as another word. This phenomenon is normally referred to as the out-of-vocabulary (OOV) problem. The training documents for this thesis are not the output from a ASR system, but manually created summaries. Should the training documents be used only with the words in the ASR vocabulary, or with all words occurring in the training corpus?

Experiments were conducted to answer the above questions, which are based on the data set set02 (see Section 8.1) with SMART-ltc weighting scheme (see Section 8.3.2). Table 8.2 shows that using original word features or their stemmed features yields similar results. Likewise, eliminating stop words or OOV words does not affect the performance much. Using the original word features without any further processing yields the best performance. So further experiments will not perform the text preprocessing steps (eliminating stop words, stemming and eliminating OOV words) mentioned in this section.

Table 8.2: Comparison of different text preprocessing combinations.

original text					
without stop words		x			x
without OOV words			x		x
with stemming				x	x
F_1	**90.32**	89.96	90.20	89.96	89.81

8.3 Feature extraction

8.3.1 Vector space model

A very popular technique to extract features from text is to represent them in a vector space. A finite set of terms $\mathcal{T} = \{t_1, \ldots, t_T\}$ contains every term that appears at least once in a given set of documents (e.g. training or test documents). For every document, one T-dimensional feature vector $\mathbf{x} = (w_1, \ldots, w_T)^T$ contains the weight w_i of the i-th term t_i. Several formulae to calculate the term weight will be presented in Section 8.3.2. The expression *term* can refer to different linguistic units: words, word n-grams, character n-grams, syllables, ... Usually, words are used as terms, but Section 8.3.3 describes a new approach to combine different linguistic units.

The fact that every term is represented by a dimension of its own implies that the terms are represented as being mutually independent. This assumption, although not true, works well in practice. Similarly, the Naive Bayes classifier makes the same independence assumption, but here it is realised in the classifier design, not in the feature representation.

Another aspect of the vector space representation is that the order of the terms in a document is ignored. So a document will be projected onto a bag of unordered terms. For this reason, this representation is frequently called *bag-of-terms*[2]. The vector space representation makes it easy to compare two documents: The cosine between the two document vectors is a good measure of similarity, which is why this representation is often used in information retrieval.

8.3.2 Term weights

The following term weights are frequently used for text classification and retrieval.

[2] Actually, it is mostly called bag-of-words, since almost all approaches use words as basic text representation unit.

TF-IDF The TF-IDF weight is an old weighting scheme and was successfully used, for example for document queries [74]. The inverse document frequency $IDF(t_i)$ can be calculated as

$$IDF(t_i) = \log\left(\frac{|D|}{DF(t_i)}\right). \tag{8.1}$$

The document frequency $DF(t_i)$ is the number of documents in which term t_i occurs at least once. $|D|$ is the total number of documents. The TF-IDF term weight of term t_i in document d is

$$TFIDF(t_i, d) = TF(t_i, d) \cdot IDF(t_i). \tag{8.2}$$

$TF(t_i, d)$ is the term frequency of term t_i in document d, i.e. the number of times t_i occurs in d. The TF-IDF measure gives a high weight to terms that occur many times in document d, applying the assumption that if a word repeats, it is important for that document. The role of the IDF component is to reduce the weight for terms that appear uniformly throughout the document collection. Such terms can hardly help to distinguish between different topics. IDF is log-smoothed to prevent terms that appear in only a few documents from receiving very high TF-IDF scores.

ITF The Inverse Term Frequency (ITF) is defined as [58]

$$ITF(t_i, d) = 1 - \frac{r}{TF(t_i, d) + r} \tag{8.3}$$

with usually $r = 1$.

SMART-ltc According to Buckley et al. [15], the term weight $WT_{\text{norm}}(t_i, d)$ shows good results in practice:

$$\begin{aligned} TF_{\text{new}}(t_i, d) &= \log\left(TF(t_i, d) + 1\right) \\ WT_{\text{new}}(t_i, d) &= TF_{\text{new}}(t_i, d) \cdot \log\left(\frac{|D|}{DF(t_i)}\right) \\ WT_{\text{norm}}(t_i, d) &= \frac{WT_{\text{new}}(t_i, d)}{\sqrt{\sum_j \left(WT_{\text{new}}(t_j, d)\right)^2}}. \end{aligned} \tag{8.4}$$

(8.4) was used for the SMART retrieval system and is called “ltc” weighting in [15], so in this thesis, it referred to as the SMART-ltc weighting.

It has been defined in the literature whether the document corpus that is used to calculate the above term weights is the training set, the test set, or the union of both. Hence, experiments

were conducted with all three alternatives. It turned out that the best results (measured with F_1, see Section 4.1 for the definition of evaluation measures) were obtained when both the training and the test features are calculated using the training data as the underlying corpus.

The performance of the above mentioned term weights is discussed in Section 8.6.1.

8.3.3 Combination of different linguistic units

According to T. Joachims, ” a high level of redundancy is a desirable property of text-classification tasks” [43]. This can be explained by the fact that it was observed in many pattern recognition applications that the exploitation of different information sources for the same recognition task often leads to different errors in the recognition results, which are very often complementary. This means that an appropriate exploitation of these sources can effectively reduce the error rate. Thus, the strategy of using such redundant information sources in combination with Support Vector Machines was chosen.

The state of the art in Information Retrieval research is to represent texts by one type of terms. Normally, words are used as terms, but there are also approaches which use syllables, phonemes [58], character n-grams and word n-grams. These types of terms, which we refer to as linguistic units, are always used exclusively. However, from a theoretical point of view, it should be promising to represent documents by more than one linguistic unit. This will not only add redundancy, which is favourable for the above reasons, but at the same time it adds some new information which could help the topic identification process.

Incorporation of several linguistic units will lead to a significant increase in the size of the vocabulary, because it will, for example, not only consist of all words in the text corpus, but also of all word 3-grams. Large feature vectors are indeed a handicap for many classification algorithms, because high-dimensional features will result in complex decision boundaries between classes; the classifier is thus likely to overfit to the training data and cannot generalise well. This is also known as the curse of dimensionality [10]: the number of training examples has to grow exponentially with the feature dimension if the quality of the model is to remain constant. SVMs, however, do not suffer from the curse of dimensionality. The complexity of their decision boundaries (measured by the Vapnik-Chervonenkis dimension) is independent of the feature dimension. SVMs are therefore the most promising classifier in conjunction with the different linguistic unit representation.

Two approaches were used to make use of different linguistic units. The first one concatenates their different text representations, and then forms one feature vector from the combined text (see Figure 8.1 a). The second one creates separate feature sub-vectors for every text representation, and afterwards concatenates the feature vectors to one big feature vector (Figure 8.1 b).

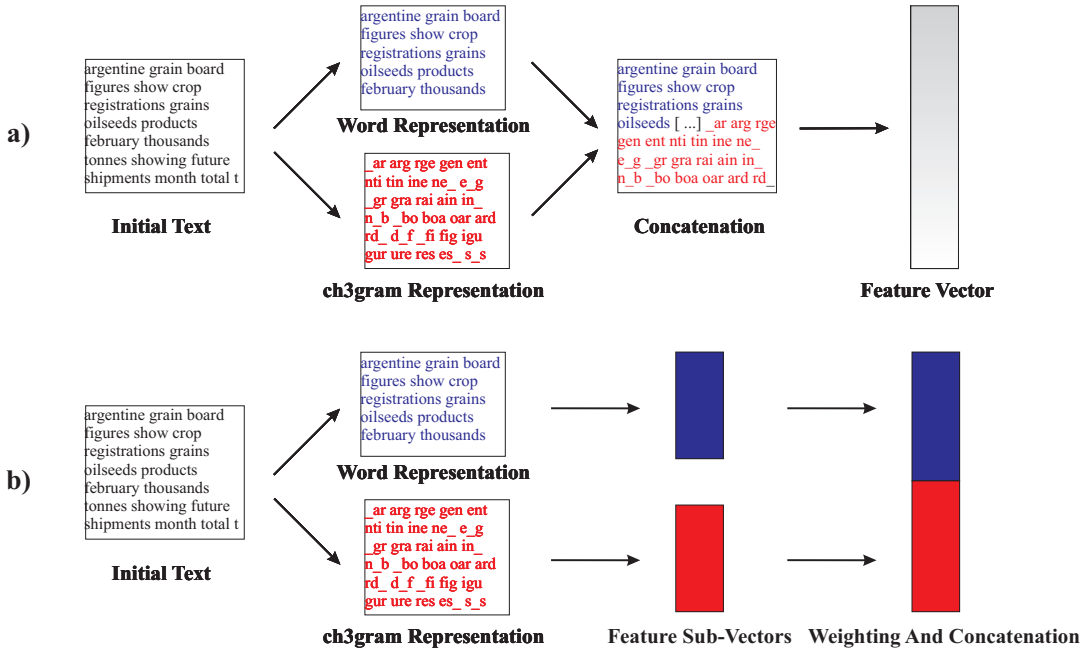


Figure 8.1: Two methods of feature vector creation for SVM experiments. The text inside the boxes was taken from the Reuters corpus evaluated in Section 8.6.6.

The following linguistic units are used [19]:

- words,
- character 3-grams (abbreviated ch3gram),
- soundex.

The soundex representation was originally developed for the field of genealogy to encode names so that similar sounding, but differently spelled names will have the same code. For example, the names *Smith* and *Smythe* have the same code S530. For English, the soundex coding algorithm [102] is:

- All codes begin with the first letter of the word followed by a three-digit code that represents the first three remaining consonants. Zeros will be added to words that do not have enough letters to be coded.
- The letters with similar sounding are coded with the same number:
  - B, P, F, V $\rightarrow$ 1,
  - C, S, G, J, K, Q, X, Z $\rightarrow$ 2,
  - D, T $\rightarrow$ 3,
  - L $\rightarrow$ 4,
  - M, N $\rightarrow$ 5,
  - R $\rightarrow$ 6.

- The letters A,E,I,O,U,Y,H and W are not coded.
- Words with adjacent letters having the same equivalent number are coded as one letter with a single number.

Such a sounding coding is a good idea for spoken document classification in order to decrease the effects of speech recognition errors.

The classification results of this novel feature representation approach are presented and discussed in Section 8.6.1

8.4 Choice of C using cross-validation

To identify the best parameter setting for the SVM misclassification tolerance parameter C (Section 3.3), 10-fold cross validation was carried out (see Algorithm 8.1 and [23]). The training set is randomly split into $n = 10$ parts of equal size, where $n - 1$ parts are used for training a classifier which is then tested on the remaining part. This procedure is repeated n times, so that every part is tested once. The classification accuracy of such a run (where C is fixed) is computed, and the whole run is repeated for different values of C. The best setting of C is then assumed to be the one that produced the run with the highest accuracy.

Creating a random split is not straightforward. One cannot sequentially assign every story to a (pseudo-)randomly chosen part, because this will create parts with a highly unbalanced number of stories. A better approach is to swap every story with a randomly chosen story that appears after it, and then creating each part from n/N consecutive stories (where N is the total number of training stories).

Two training sets were used to determine the best choice for C: set02 and set03. To identify the region where the optimal C might be, a first, coarse run was performed with C varying on a large scale:

$$C \in \{2^{-5}, 2^{-3}, 2^{-1}, 1, 2^{1}, 2^{3}, ..., 2^{15}\}.$$

The resulting accuracies are depicted in Figure 8.2. As the relevant regions of maximal accuracy cannot be identified here, a part of this figure is zoomed out and displayed in Figure 8.3 (a) for set03 and (b) for set02 (dotted lines). A finer run was then performed to find a more precise value of C, using

$$C \in \{1, 1.3, 1.7, \ldots, 16, 20, \ldots, 384, 512\}.$$

Algorithm 8.1 n-fold cross-validation to get optimal setting of C.

Define the set $\mathcal{C}$ over which the parameter C will run.
Split the training set into n equal partitions.
for $C \in \mathcal{C}$
  $r := 0$
  for $i := 1$ **to** n go through all partitions i
    Classify partition i using system trained on the remaining $n - 1$ partitions.
    $r := r +$ *number of correctly classified samples*
  end
  $\textit{overall\_accuracy[C]} = \dfrac{r}{\textit{number of documents in whole training set}}$
end
The best setting of C is
$C := \text{argmax}_C$ *overall_accuracy[C]*.

The results from the fine run are also depicted in Figure 8.3 (solid line). The optimal value of C for set02 lies around 1.3, for set03 it lies around 1.7. For all experiments, a value of $C = 1.3$ was chosen.

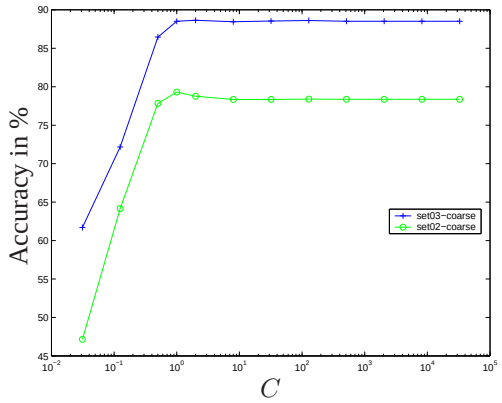
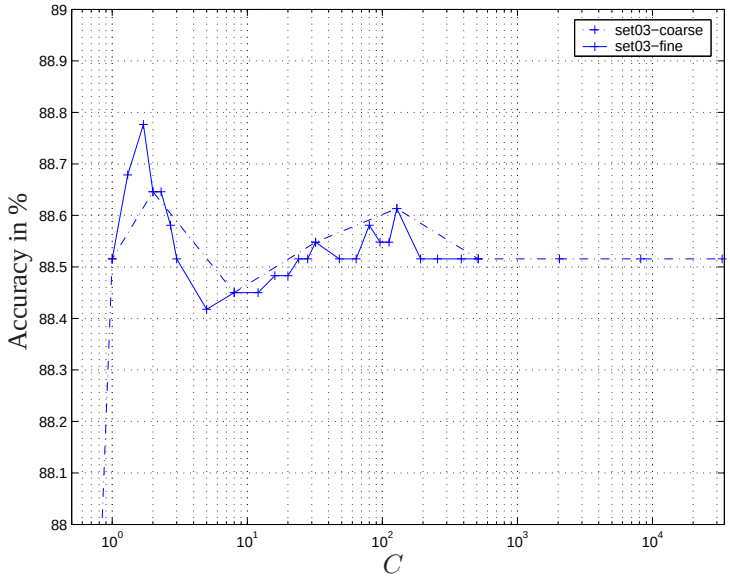


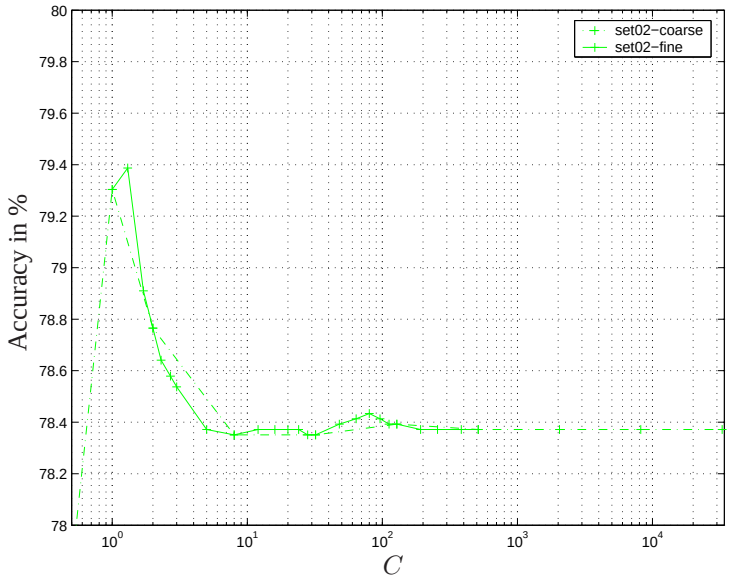
Figure 8.2: Accuracies from 10-fold cross validation vs. C (coarse run).

8.5 Character n-gram size and choice of kernel

For a character-based document representation, the optimal size n of the n-grams has to be determined. Mayfield [53] determined that character 6-grams yield the best performance. Our experiments, however, indicate that $n = 3$ is best (determined using ITF term weighting). Leopold et al. [47] have observed that the best length of one term (in their case, phonemes) depends on the size of the class. Smaller classes are better represented with shorter term units.



(a) Set03



(b) Set02

Figure 8.3: Accuracy vs. C of the coarse (from Figure 8.2, dotted lines) and the fine cross-validation run (relevant part only).

This corresponds to the fact that most training classes used for the experiments of this thesis are small.

Furthermore, the correct SVM kernel has to be chosen. Initial tests showed that for non-probabilistic SVMs, a linear kernel is a good choice. Other kernels perform equally good or worse. This also supports the findings by others [41] that text document classification is usually a linearly separable task.

8.6 Experiments

8.6.1 Choice of term weighting scheme

A preliminary set of experiments was conducted to find out which term weighting scheme best represents the documents (in terms of best classification result). The documents were represented by word or by character 3-grams (ch3gram) alone, and by the combination of both representations in order to see the effect of the combination approach (see Table 8.3, where the best text representation for every experiment is indicated by a bold figure). The texts based on words and on ch3grams were concatenated, and then the feature vectors were built.

Table 8.3: Results (F_1 measure) with different weighing schemes.

Data Set	word	ch3gram	word+ch3gram
	TF-IDF weighting		
set02	82.8	89.5	**89.9**
set03	93.3	94.0	**94.8**
set04	**80.1**	79.4	79.8
	ITF weighting		
set02	88.6	89.7	**90.9**
set03	94.1	95.4	**96.2**
set04	79.3	79.7	**82.0**
	SMART-ltc weighting		
set02	90.3	90.6	**90.7**
set03	95.7	**96.1**	**96.1**
set04	88.0	87.0	**88.2**

Two conclusions can be drawn from these experiments: first, the best weighting scheme for all three tests is SMART-ltc. Second, the concatenation of different linguistic units yields, in almost all cases, better results compared to using one linguistic unit alone. However, the improvements are very small (about 1% absolute only).

A soundex representation does not help to improve the results (see Table 8.4). Using soundex as the sole representation yields worse performance that using word or ch3gram alone. Combining soundex and 3gram representation is better than ch3gram alone, but still worse than the combination of words and ch3grams.

Table 8.4: Results (F_1 measure) for soundex representation with SMART-ltc weighing.

Data Set	sdx	sdx+ch3gram	word+ch3gram
set02	88.4	90.2	**90.7**
set03	94.4	**96.1**	**96.1**
set04	82.2	87.6	**88.2**

8.6.2 Weighting of linguistic units

Different linguistic units will contribute to the classification result to a different degree. It should therefore be favourable to weight more important units [19]. As this cannot be accomplished with the feature extraction according to Figure 8.1 a, the feature vectors have now to be created separately from the different text representations, then multiplied by a constant weighting coefficient, and finally be concatenated (see Figure 8.1 b). Results obtained with the first method, and with the second method with *equal* unit weighting, are nearly identical (they differ only between 0 % and 0.5 %, depending on the data set[3]). Thus the results from the first and the second method can be compared with each other.

The results for different word weights (the weight of character 3-grams was fixed at 1) are included in Figures 8.4 to 8.6 (*decval* graph). The word weight which leads to the best F_1 measure is listed in Table 8.5 for each data set. In almost all cases, word weights not equal to 1.0 perform best, but the increase in performance, compared to $w_{\text{word}} = 1.0$, is usually not very high. The best word weights are in the range between 0.8 and 3.0, usually greater than 1.0. One exception is the macro-averaged F_1 measure for set04 with its best value at $w_{\text{word}} = 10$. Hence, one can conclude that words carry more information about topics than character 3-grams, but the contribution of 3-grams must not be neglected.

Table 8.5: Best F_1 measures for different word weights w_{word}, obtained with voted SVM.

	micro-avg		macro-avg	
Data Set	max. F_1	at w_{word}	max. F_1	at w_{word}
set02a	90.7	1.5	39.2	1.5
set03	96.1	0.8, 1.0	71.2	3.0
set04	88.1	2, 3	37.0	10

One striking fact is that classification performance decreases only slightly when switching from set02a (where the classifier has to choose among 22 classes) to set04 with its 827 training classes.

[3] For micro-averaged F_1 measures.

8.6.3 Couplers for non-probabilistic SVMs

As already described in Section 3.5, SVMs are binary classifiers, i.e. they can only distinguish between two classes. For multi-class categorisation, the one-against-one approach (see Section 3.5) was applied. For non-probabilistic SVMs, Section 3.5 presents the standard voting coupler and introduces the new decision value and distance couplers that are used to combine the $K(K-1)/2$ binary predictions.

The performance of these couplers on set02a and set03 was evaluated (see also [34]). The corresponding F_1 measures are depicted in Figures 8.4 and 8.5.

One problem arose when computing the distances based on the classifiers of set04 was that some (259 out of $341,551 \hat{=} 0.1\%$) binary SVM models had values of $\|\mathbf{w}\| \approx 0$. The distance to the hyperplane of such a model will thus become very large and distort the results. Indeed, the distance coupler always predicted one single class, independently of the test document, but dependent on the word weight vector w_{word}. Typical, non-distorted summed distances lie around 250, while distorted sums are about $8 \cdot 10^8$! There are three approaches to tackle the problem:

1. Very small $\|\mathbf{w}\|$ indicate that some class pairs cannot be separated very well. Choosing a kernel other than the linear one might help to improve the performance.

2. An analysis of the binary classifiers and the training document frequencies revealed that 261 out of 827 training classes have at least one binary model with a "singular" value of $\|\mathbf{w}\| \approx 0$. 181 out of these 261 classes consist of three or less training documents. This indicates that training topics with a very low number of associated documents are the dominant cause for the observed numerical problems. Excluding documents whose topic appears very infrequent should help. The disadvantage of this approach is that it decreases the number of training classes, i.e. the number of topics that can be potentially detected.

3. Ignore the distance results for the models with $\|\mathbf{w}\| \approx 0$. This approach, just as the first one, does not not require reduction of the number of training topics.

It is difficult to judge a-priori which of the three approaches is best. For the first algorithm, a RBF kernel was chosen with the standard parameters of `libsvm`; this kernel performs well for probabilistic SVM. It however turned out that this approach cannot solve the problem at hand.

To analyse the second approach, a new training set was created that contains only those topics that appear in at least 4 documents. It covers 560 topics in 20,736 documents. The test set remained the same. The results on this test set were disappointing: just as in set04, only

one single topic was predicted for all test stories. Further tests with a decreasing number of training topics were not performed, as only a significant reduction of topics seems promising.

The third approach revealed to be the best one; the performance values seem virtually unaffected by ignoring (i.e. not testing on) binary models with $\|\mathbf{w}\| \approx 0$ (see Figure 8.6).

The reason that $\|\mathbf{w}\| \approx 0$ is that there are several features in the training set that are assigned to more than one class (by being separately listed with different class labels). This fact was observed at a late stage when most experiments were already completed; as ignoring models with zero $\|\mathbf{w}\|$ doesn't change the results, no experiments with a new training set were conducted.

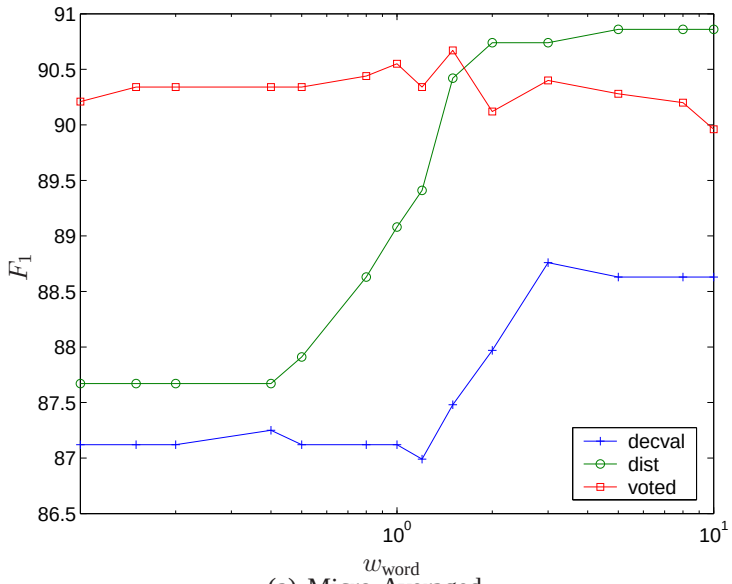
When looking at the results of all data sets, one can see that the voting coupler is usually the best one, both for micro and macro averaging. One exception is the micro-averaged results of set02a: For high w_{word}, the distance coupler outperforms the voting coupler. The two new couplers *distance* and *decision value* tend to perform better with higher w_{word}, one exception being the micro-averaged results of set04. The *voting* coupler is only slightly affected by changing the word weight.

8.6.4 Couplers for probabilistic SVMs

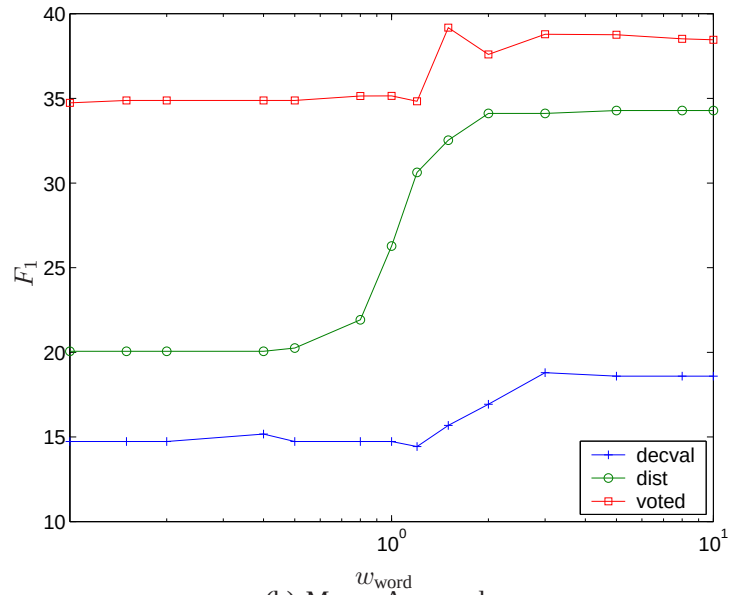
Experiments using probabilistic SVMs could only be performed with the data sets set02a[4] and set03 (see also [34]). The training set of set04 contains several classes with only very few training stories assigned, and can therefore not be used to reliably estimate the sigmoid parameters A, B using cross-validation. Experiments on set02 and set03 were conducted without using cross-validation, but the results significantly differ from those obtained with cross-validation. As theory (see Section 3.5.2) suggests that cross-validation is the correct way, experiments on set04 were discarded.

Two SVM kernels were tested for the seven couplers for probabilistic SVMs described in Section 3.5.2: linear and RBF. For the RBF kernel, the standard parameters of `libsvm` were used. The sigmoid parameters A, B were estimated by five-fold cross-validation of the training set. The results for the RBF kernel are depicted in Figures 8.7 and 8.8. For set02a and micro-averaged F_1 measures, the overall best coupler is pkpd. Vote-probwght2 is the best for large values of w_{word}, but cannot outperform pkpd. On set03, vote-probwght1 performs best, but it is not on top for *all* w_{word}. With macro-averaging, pkpd (set02a) and minpair, markov, and pkpd (set03) show peak performance.

[4] Cross-validation cannot be performed on set02, because it contains a topic with only two stories. In set02, these two stories are removed.



(a) Micro-Averaged

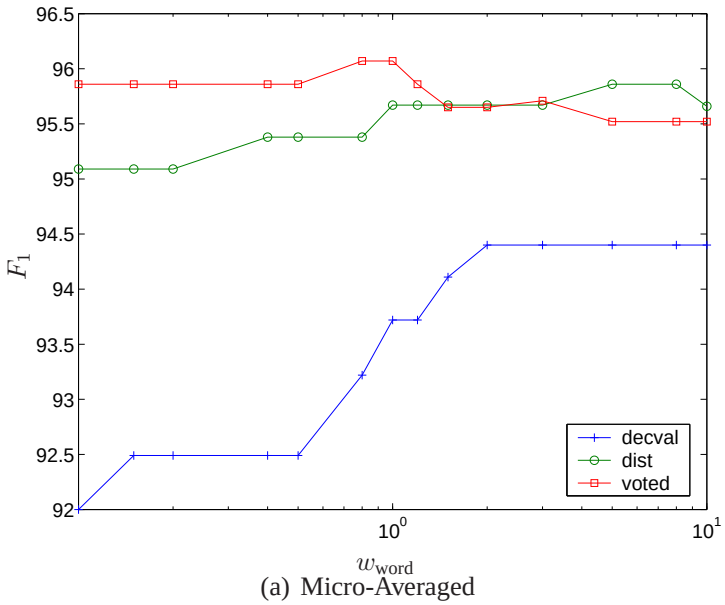


(b) Macro-Averaged

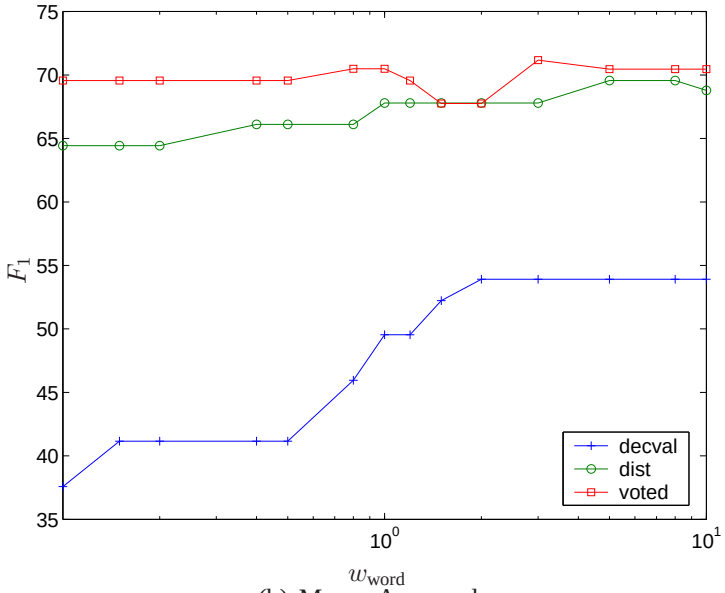
Figure 8.4: Performance of three non-probabilistic SVM couplers on set02a.

One disadvantage of the voting coupler is that it sometimes (e.g. in 18 out of 443 test documents) produces the same posterior probability for more than one class. Its class prediction is therefore sometimes ambiguous. For the tests presented here, one out of the tie classes is chosen at random.

The linear kernel performed worse than the RBF kernel. This is quite an interesting result, because for non-probabilistic SVMs, the best kernel was the linear one. To our knowledge, the



(a) Micro-Averaged

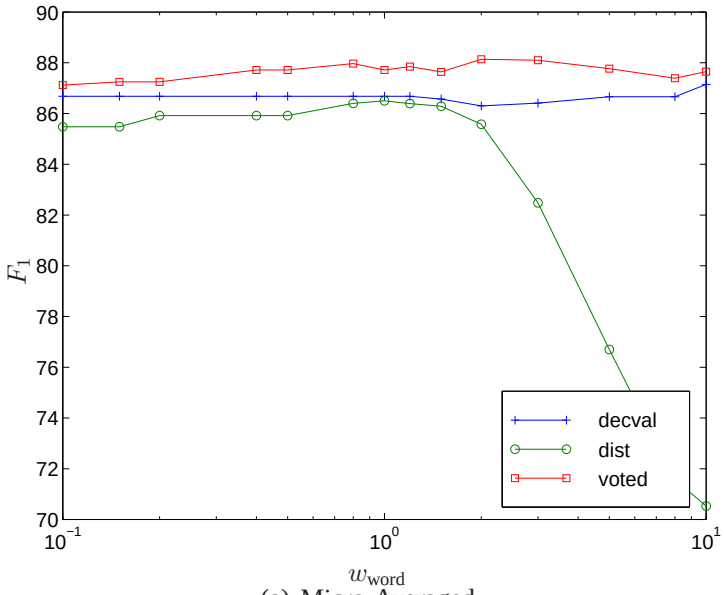


(b) Macro-Averaged

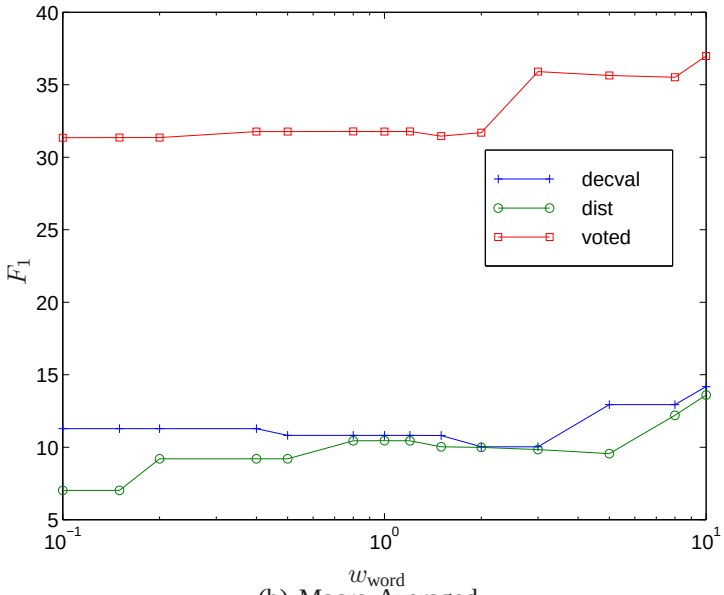
Figure 8.5: Performance of three non-probabilistic SVM couplers on set03.

fact the best kernel changes when one changes from npSVM to pSVM was not yet reported in the literature. The detailed results for this kernel are not depicted here, but are included in Table 8.6.

It is interesting to note that for npSVM couplers, the performance increases with w_{word} nearly monotonically, but the same is not true for pSVMs. Here, the F_1 measures tend to go up and down. This may be a consequence of the biased estimation of the posteriors: with high-



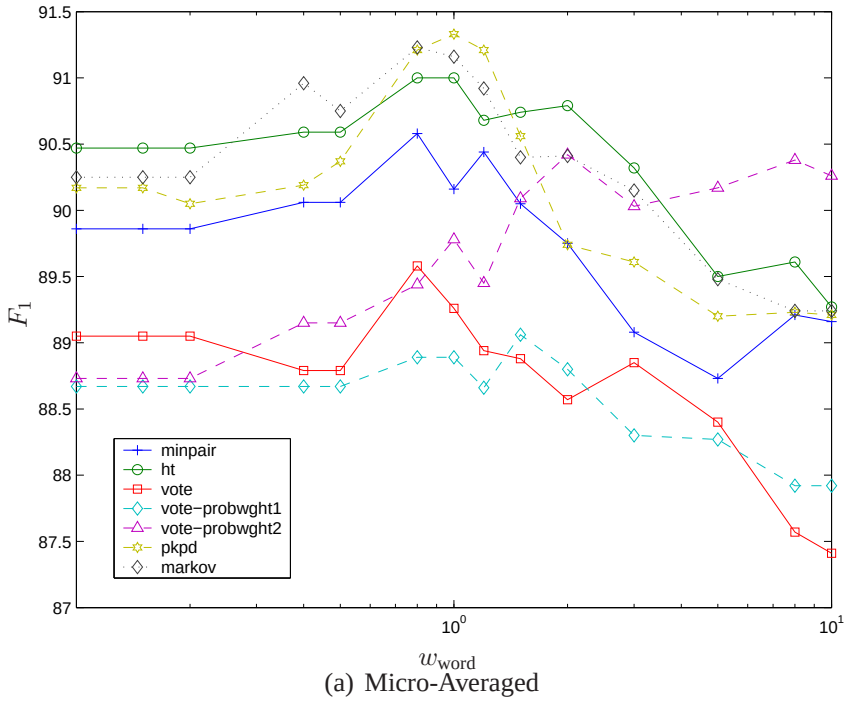
(a) Micro-Averaged



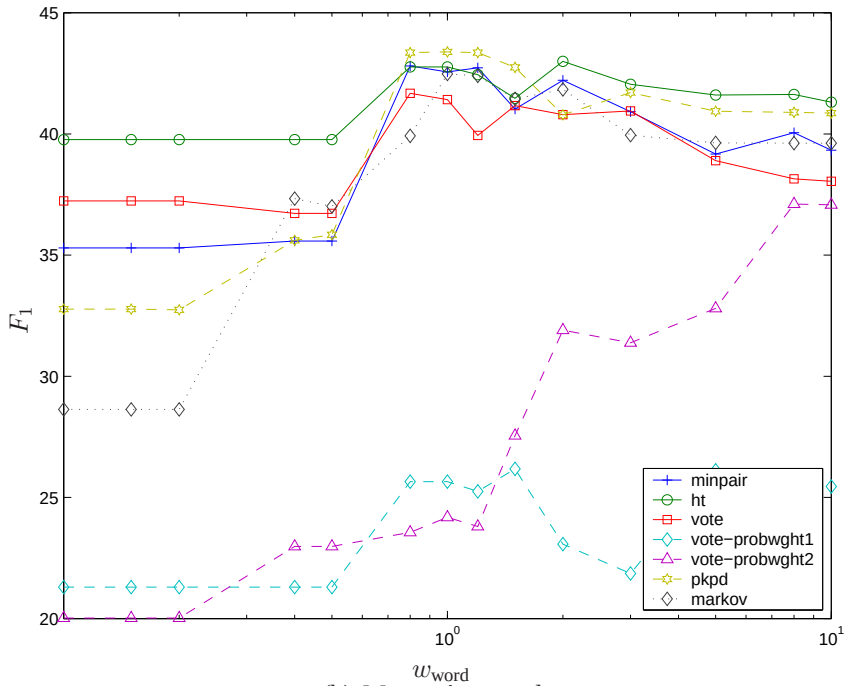
(b) Macro-Averaged

Figure 8.6: Performance of three non-probabilistic SVM couplers on set04.

dimensional features, but relatively few training examples, many features will lie on the margin, i.e. $g(\mathbf{x}) = 1$, so the estimation will be biased towards the margin [62]. Cross-validation to estimate the sigmoid parameters A, B can weaken, but not remove this effect.

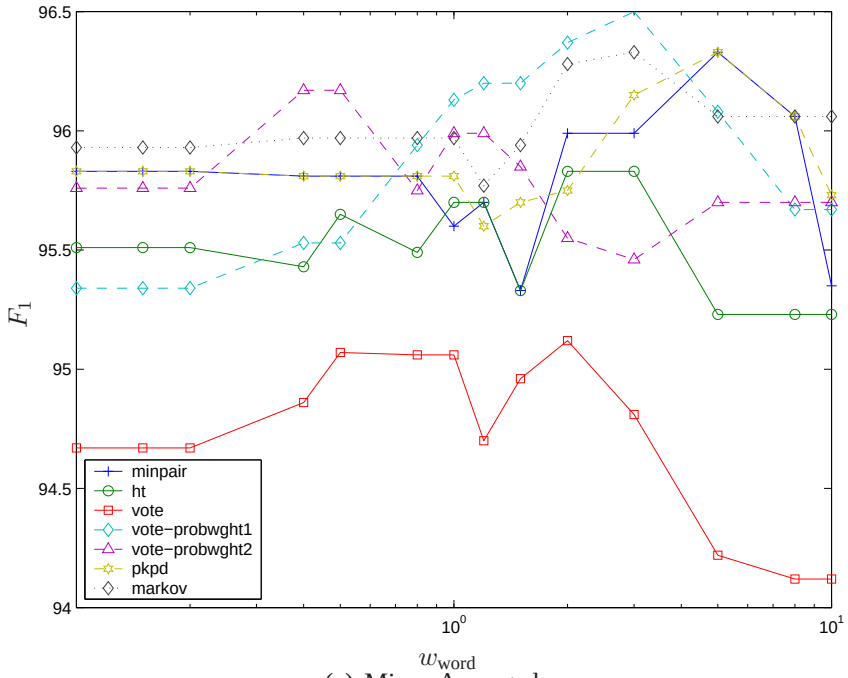


(a) Micro-Averaged

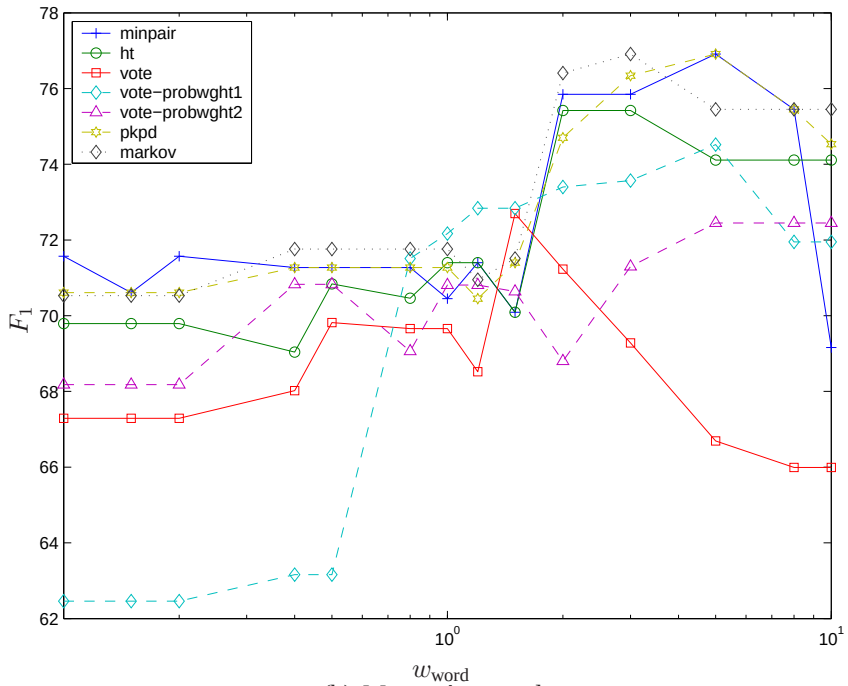


(b) Macro-Averaged

Figure 8.7: Performance of seven probabilistic SVM couplers with RBF kernel on set02a.



(a) Micro-Averaged



(b) Macro-Averaged

Figure 8.8: Performance of seven probabilistic SVM couplers with RBF kernel on set03.

Table 8.6: Summary of best F_1 results for SVM classifiers (vp1,2 = vote-probwght1,2). The best result of each line is printed in bold.

	npSVM linear kernel		**pSVM RBF kernel**		**pSVM linear kernel**	
averaging	**coupler**	F_1	**coupler**	F_1	**coupler**	F_1
	set02a					
micro	distance	90.9	pkpd	**91.3**	minpair	90.9
macro	voted	39.2	pkpd	43.4	minpair	**43.7**
	set03					
micro	voted	96.1	vp1	**96.5**	ht,markov,vp2	96.1
macro	voted	71.2	minpair, markov, pkpd	**76.9**	ht	73.6
	set04					
micro	voted	88.1	—	—	—	—
macro	voted	37.0	—	—	—	—

8.6.5 Statistical tests for significant difference of classifiers

Although the coupler with maximum F_1 value can be clearly identified, there are several couplers with nearly the same performance. The question arises whether the couplers' difference in performance is statistically significant. One widely accepted method to test for significant difference is McNemar's test [22, 62] (also known as χ^2 test): The χ^2 statistics is computed for every coupler pair (with fixed w_{word}). χ^2 depends on the number of test stories in which two couplers agree or disagree (see Appendix A for details). If the computed value $\hat{\chi}^2$, which is an estimate for the real, but unknown value χ^2, is greater than 3.8, the two couplers can be assumed to be significantly different; this assumption is made with an error rate of 5%. Note that in order to calculate the χ^2 statistics, one counts the overall number of correct or incorrect predictions, and not a per-topic average. This corresponds to the micro-averaged precision measure, whereas the evaluation measure used here, F_1, is the harmonic mean of precision and recall. Hence, there are cases where the difference of F_1 between two couplers is low, but χ^2 is high, and vice versa. Even more, cases were observed where one coupler had a higher F_1 value, but the χ^2 statistics suggested that the other one was better.

McNemar's test was applied to the predictions of conventional SVMs, and to probabilistic pSVMs with RBF kernel. For every w_{word}, the χ^2 statistics between all coupler pairs was computed. For the w_{word} at which the best performance was observed (cf. Table 8.6), Table 8.7 lists those pSVM couplers that are significantly worse than the best performing coupler. It also notes the corresponding $\hat{\chi}^2$ value. The best coupler is significantly better than only one or two other couplers. In other words, it is not significantly different from four or five out of seven couplers. Among the conventional SVM couplers, the decision value coupler performed

significantly worse than the distance and the voting coupler. The latter two do not differ significantly.

Taken together with the fact that there is no coupler that is always best, and that the best coupler changes frequently when varying w_{word} (see Figures 8.7 and 8.8), it can be concluded that the correct choice of a probabilistic coupler for spoken document classification is difficult. For every setting (dataset, classifier parameters), another coupler might be better, but our results indicate that the difference in performance between the couplers is not necessarily significant. The new couplers vote-probweight1, 2, and distance have the advantage that they are easy to implement and have low run-time requirements. As for the pSVM couplers, this is also true for the voting coupler. But the voting coupler will often lead to ties in class prediction, which will almost never happen for the new couplers.

These facts indicate that the new couplers, except for the decision value coupler, should be preferred over the other known couplers.

Table 8.7: Significantly different couplers according to McNemar's test. Results are listed only for pSVM with RBF kernel. For every data set, the w_{word} was chosen that yielded the best micro-averaged F_1 measure.

data set	w_{word}	the best coupler ...	is significantly better than ($\hat{\chi}^2$) ...
set02a	1.0	pkpd	minpair (6.1), vote (10.6)
set03	3.0	vp1	vote (5.9)

8.6.6 Experiments on the Reuters data set

To see whether the findings of this chapter hold also on a different, widely used data set, the most important experiments were repeated on an English Reuters corpus. It was widely used for text categorisation by many researchers, which makes it one of the most favourite standard test collections. The set consists of the ten most frequent categories of the Reuters-21578 corpus [2], ApteMod version. Although it does not contain spoken text, but newswire text, it can nevertheless be used for comparison, as classification of spoken documents and written text are not very different. 5915 text documents were used for training and 2307 for testing.

The best F_1 measures on the Reuters data set for conventional SVMS, and for pSVMs with linear and RBF kernel are listed in Table 8.8. The following findings were observed:

- Giving weights to the word sub-feature vectors is favourable.
- The decision value coupler is significantly worse than the voting couplers (conventional SVMs).

- The distance coupler is not statistically significantly worse than the voting coupler (when measured with micro-averaging). It is even better than the voting coupler when measured with macro-averaging.
- There is no best coupler for pSVMs. The performance of most couplers, including the two new couplers, does not differ significantly.
- The best F_1 measures were produced by pSVMs, not by conventional SVMs.
- In contrast to the findings on the German data set, the best kernel for pSVMs was the linear kernel, just like the best kernel for conventional SVMs. This indicates that the fact that the best pSVM kernel on the German data set was the RBF kernel does depend on the data set and is not a general property.

Thus, all main findings of this chapter could be reproduced on the Reuters set.

Table 8.8: Summary of best F_1 results for SVM classifiers on the Reuters data set (vp1,2 = vote-probwght1,2). The best result of each line is printed in bold.

	npSVM linear kernel		**pSVM RBF kernel**		**pSVM linear kernel**	
averaging	**coupler**	F_1	**coupler**	F_1	**coupler**	F_1
micro	voted	94.2	vp2	92.5	minpair; markov; pkpd	**94.5**
macro	dist	86.1	vp2	84.6	pkpd	**88.2**

8.7 Conclusion

For the German data sets, Table 8.6 lists the best couplers of conventional SVMs with linear kernel, and probabilistic SVMs with linear and RBF kernels. Both micro and macro averaged results are included. The best result of each line is printed in bold figures in order to be able to compare npSVMs and pSVMs at one glance. In all cases, the probabilistic SVMs yield the best results. In three of four cases, this was accomplished with the RBF kernel, in one case with the linear kernel. For conventional SVMs, the voting coupler is usually the best. For pSVMs, there is no single best coupler that performs significantly better than the other ones. As explained above, the probabilistic SVMs could not be used for set04 because it contains a lot of training topics with very few stories each.

For the first time, the performance of conventional and probabilistic couplers on text classification is investigated. The main findings of this chapter can be summarised as follows:

- Among the examined term weighting schemes, SMART-ltc showed the best performance. This scheme was also found to be the best for topic clustering for topic detection [7, p. 11].
- The combination of different linguistic units into one feature vector improves the classification rate. Giving more weight to the word sub-vectors is favourable.
- Among the couplers tested on non-probabilistic SVMs, the voted coupler is usually the best one. The incorporation of confidence, as implemented by the new distance and decision value couplers, surprisingly did not help to increase the classifier performance. On the other hand, the distance coupler does not perform significantly worse than the voting coupler.
- For probabilistic SVMs, there is no best coupler. Different settings bring different best couplers.
- The newly introduced couplers *distance*, *vote-probweight 1* and *vote-probweight 2* do not perform significantly different than the other couplers discussed here. The advantage of the probweight couplers over the other couplers is that they are easy to implement, have low run-time requirements, and will not predict more than one correct class (ties).
- The best classifier parameter settings are found among the probabilistic SVMs, in most cases using the RBF kernel (German data) or the linear kernel (English data set).
- The best kernel for conventional SVMs (German and English data) and for pSVMs (English data only) is the linear kernel. For npSVMs and the German data, the RBF kernel is usually a better choice. This latter result is quite interesting, because text classification is usually thought to be a linearly separable task, and almost always linear kernels are used (both for non-probabilistic and for probabilistic [24] SVMs).

Chapter 9

Overall system performance

The experiments with the topic classifiers described in the preceding chapters dealt with optimal topic boundaries only. While this setting is ideal for comparing and analysing the topic classifiers alone, it does not tell anything about the performance of the combined system of speech recognition, topic segmentation and topic classification.

This chapter presents experiments on the whole system. Thus, it also answers the question whether the errors of the topic segmentation module have much impact on the topic classification rate. If the segmentation did only insert, but never omit boundaries, there might be a good chance that the classifier will nevertheless perform well. However, the segmenter also omits true boundaries, so that the topic classifier's performance will definitely degrade.

This chapter analyses two topic classifiers: Naive Bayes and SVM. The hybrid HMM/NN classifier is omitted because it already turned out that it is worse than the Naive Bayes (see Section 7.3). The presented results include the performance of the topic classifiers both with manually and with automatically topic segmented test data.

9.1 Experiments with Naive Bayes

The Naive Bayes approach (Section 7.2) was implemented without a background model for off-topic stories. The *Off-topic* label is assigned to a story if the classifier's prediction has a confidence value below a pre-defined threshold (see Section 7.2.3 for details).

The training and test set used for Naive Bayes experiments is set01. As usual, the training data consists of manually written summaries of TV and radio broadcasts which are topic-labelled. There are 1051 different topics in 154,143 training summaries. The test data consists of ASR transcriptions of TV news shows created by the ASR module. It is the same data that was used for all SVM experiments (see Section 8.1), except for stemming and stop word removal. Both

the training and the test data were stemmed, and stop words were removed. The tool used for stemming is a Perl script by the University of Dortmund [61]. The stop word list was manually created and contains 149 entries.

Two stories in the test set contain a topic that was not trained, and consequently can never be detected.

For data set01 with the baseline HMM classifier system, the confidence threshold was varied in the range of $[0.5, \ldots, 0.98]$. For every threshold value, the F_1 measure (micro and macro averaged) was computed. The results are depicted in Figures 9.1 (a) and (b). Sub-figure (a) shows the performance on a test set that was created with manually inserted (i.e. "correct", or reference) topic boundaries. The topic boundaries of the test set treated in sub-figure (b) were created with the topic segmentation system presented in chapter 6.

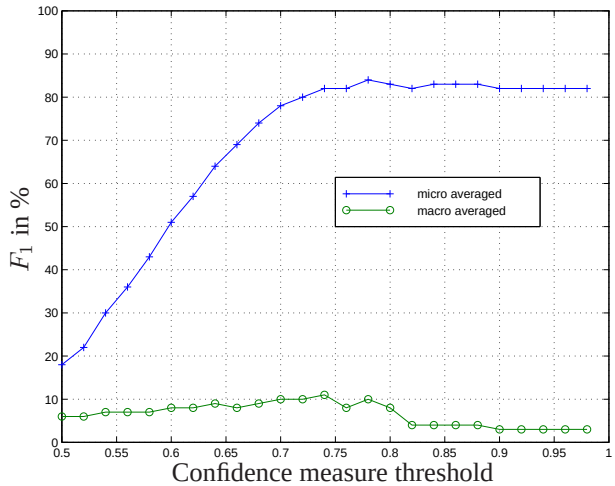
The best combination of F_1 micro and macro averaged values is listed in Table 9.1. For both test sets, with automatically and with manually created topic boundaries, the confidence threshold can be set to 0.78.

Considering the fact that the classifier had to choose among 1051 topics, the micro averaged value of 83 % for the automatic topic boundary set is quite good. However, the macro averaged value of 6 % is disappointing.

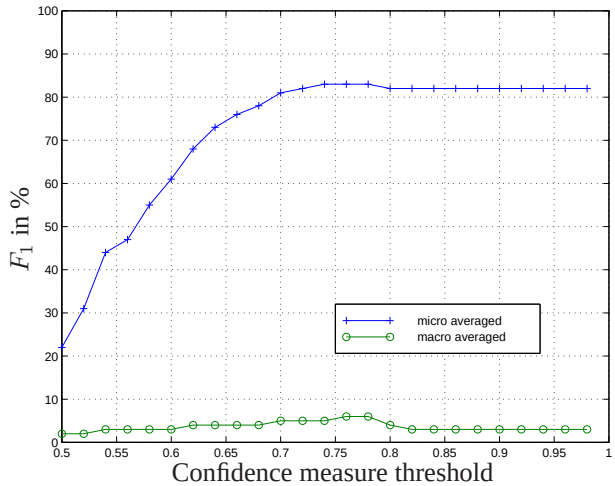
In the experiment setting presented above, the *Off-topic* label was merely assigned by rejection based on the confidence value. A different experiment was conducted without a confidence measure, but where the classifier could directly assign the *Off-topic* label because a model was trained for it. Results, however, were worse than those with confidence.

Table 9.1: Classification results of the baseline system on the data set01 for automatically and for manually segmented topic boundaries.

topic segmentation	F_1 **micro averaged**	F_1 **macro averaged**	**at confidence value**
SVM			
manual	88	37	–
automatic	86	16	–
Naive Bayes			
manual	84	10	0.78
automatic	83	6	0.76 and 0.78



(a) Manual topic segmentation



(b) Automatic topic segmentation

Figure 9.1: Classification performance (measured by F_1) vs. varying confidence measure threshold for the baseline system. Data set is set01, with manually (a) and automatically (b) segmented topic boundaries.

9.2 Experiments with Support Vector Machines

For the SVM classifier, the training and test data has to be represented in the vector space model according to the feature extraction method presented in Section 8.3. However, for the training set of set01, this results in a total feature file size of 632 MB. The training of SVMs based on such a big data set is very time consuming. It becomes even infeasible to train if probabilistic SVMs are to be trained, because of the cross-validation needed to estimate the sigmoid function (3.33). For set01, such training takes several days.

Therefore, it was decided to take a data set that is comparable to set01, but has less training stores. The data set that corresponds best to set01 (which is used for the Naive Bayes evaluation) is set04. It has got 827 different training topics in 21,376 training stories (see Section 8.1). The smaller number of training stories per topic does not mean that the training of the SVMs is less accurate (compared to Naive Bayes), since SVMs need fewer training examples than the generative models of Naive Bayes. This latter fact is due to the fact that SVMs only estimate the decision plane between two classes, and not class distributions, and also because SVMs do not suffer from the curse of dimensionality.

The SVM performance on data set set02a with its 22 topics compared to set04 does not change in the same way as the increase in topic number would suggest. Therefore, one can safely assume that the performance of SVMs on a 1051 topic set (the same number as in set01) would not be much different.

The conventional SVM (npSVM) has a classification performance of 88.1 (F_1 measure, macro-averaged) on a manually (i.e. optimally) topic-segmented set04. The corresponding macro-averaged F_1 measure is 37.0. With the same data, but now the topic boundaries of the test set being determined by the topic segmentation module presented in Section 6, the topic classifier's performance decreases: The micro-averaged F_1 measure goes down from 88.1 to 85.1, and the macro-averaged F_1 measure from 37.0 to 15.2. These figures can also be found in Table 9.1, together with their Naive Bayes counterparts.

9.3 Comparison and conclusion

When the SVM approach is compared to the Naive Bayes approach, the conclusion is that SVM performs better. It is even significantly better when comparing only the macro-averaged measures. This means that SVMs do a much better job in assigning topics to on-topic stories, i.e. to stories that are of interest to any media monitoring customer. However, especially with automatic story boundaries, the performance on on-topic stories is still unsatisfactory. The training and test set given are "real-life" data sets. Therefore, several exceptional challenges have to be taken into account when analysing the classification results:

- There are a lot of training topics (827 in set04), but the test set contains only 23 topics. This is of course realistic, as only a small subset of the topics that are to be monitored will actually appear.
- The training and the test set come from two different domains: Manually written summaries (i.e. nearly error-free text) make up the training set, while the test set is the output of an error-prone automatic speech recogniser (ASR).

- The ASR's output is not very good for reports which do not take place in a TV studio: Often, even a human cannot identify the meaning of such a transcription, because many words are mis-recognised. Only when reading the transcription and simultaneously listening to the audio, the meaning of the story becomes clear.

Given these items, the micro-averaged performance of $F_1 = 88$ is good, as is the macro-averaged value of $F_1 = 37$ (for manual topic segmentation). The latter means that of those stories that are of any interest to a media monitoring customer, about one third is correctly assigned – given the above challenges, this shows a good performance of the classifier. Even more, a 100% performance is not needed, because every prediction by the developed system would of course be cross-checked by a human.

It turns out that the topic segmentation performance has got quite a big impact on the macro-averaged topic classification rate of the whole system. The SVMs (and even more, the Naive Bayes) have problems in determining the correct topic of a story if it is split (by an imperfect topic segmentation) into two or more parts. Hence, all words of a story are needed to correctly classify it. This coincides with the findings by Joachims that most words of a story are relevant [42].

Chapter 10

Unsupervised Topic Discovery

One challenge in the media monitoring application is that the topics change rapidly. In the data sets used for this work, there were often topics that newly emerged in the test set and were therefore not present in the training set. With a fixed set of topics, such new events can of course never be detected. One remedy is to continuously update the topic models, which means retraining them at least every day. The choice of new stories used for training can be done in an unsupervised way so that no person is involved [40]. Such an approach is beyond the scope of this work, since the set of topics is assumed to be fixed.

One common drawback of all mentioned approaches is that they need a pre-defined list of topics to be detected. For the scenario of media monitoring, where the set of customers (those that will receive the media alerts) is known beforehand, a restricted topic list is useful. However, for media monitoring for e.g. private customers who do not necessarily have a fixed topic profile, it would be better to find a different approach. One option is not to provide any pre-defined topic list at all, but to derive it from the data itself. This approach is almost totally neglected in the literature, except from the work by Sreenivasa Sista et al. [80, 83, 82]. The advantage of this approach, called Unsupervised Topic Discovery (UTD), is that there is no need for a human-labelled training set, and that there is no human interaction in the finding of the topics.

The benefits of UTD for users is the assignment of a set of meaningful topic labels to each story (for example, in broadcast news), and the user can quickly browse the labels to find interesting stories instead of having to watch each story. There is no need to pre-define what the user is interested in, i.e. to define his/her topic profile. As the topics are not fixed, newly emerging topics can be easily found. In contrast to the previous chapters, where it was assumed that one story has got only one topic label, in this chapter it is assumed that usually more than one label is assigned to each story. Only the conjunction of several topic labels for one story will allow to get an idea of the contents of the story.

10.1 Overview of Unsupervised Topic Discovery

The goal of Unsupervised Topic Discovery (UTD) is to assign topic labels for each text document of a document collection without human interaction. A document can consist of plain text, for example from newspaper or news websites, or it can be the deficient output of a speech recogniser. As throughout the rest of this thesis, the terms *document* and *story* are used interchangeably.

According to Sista [82], "The unsupervised topic discovery attempts to replicate the human [topic] annotation process". In contrast to supervised topic classification, where a human annotator assigns presumably true topics to the training documents, in UTD there are nor training stories annotated by humans, nor a pre-defined list of topics which should be detected. Instead, a topic list is automatically derived from the document corpus. This step, called *Initial Topic Labelling*, also automatically annotates each document with several topics. A training process uses these topic annotations to find a statistical model for each topic, which subsequently is used for re-classifying the documents (*Final Topic Labelling*). Before Initial Topic Labelling, a *Preprocessing* step, among other things, identifies phrases in the documents. These three steps are depicted in Figure 10.1, and will be explained in detail in the following sections.

10.2 Preprocessing

The preprocessing steps follow largely the approach by Sista [82]. The following steps are performed:

- Stemming
- Phrase creation
- Stop word removal

The name identification step used by Sista, which identifies names of persons, places, and organisations, is omitted due to lack of an appropriate tool. Anyway, he reports that this step has very little effect on the final topics. The text is then stemmed using the Snowball algorithm for German [5]. Snowball is a framework that allows the creation and integration of stemmers for many languages, or as the authors put it, it "is a small string processing language designed for creating stemming algorithms for use in Information Retrieval." Many stemming rules for languages such as English (equivalent to the well-known Porter stemmer), German, Swedish, or even Finnish were created and published on the Snowball website. The German stemmer

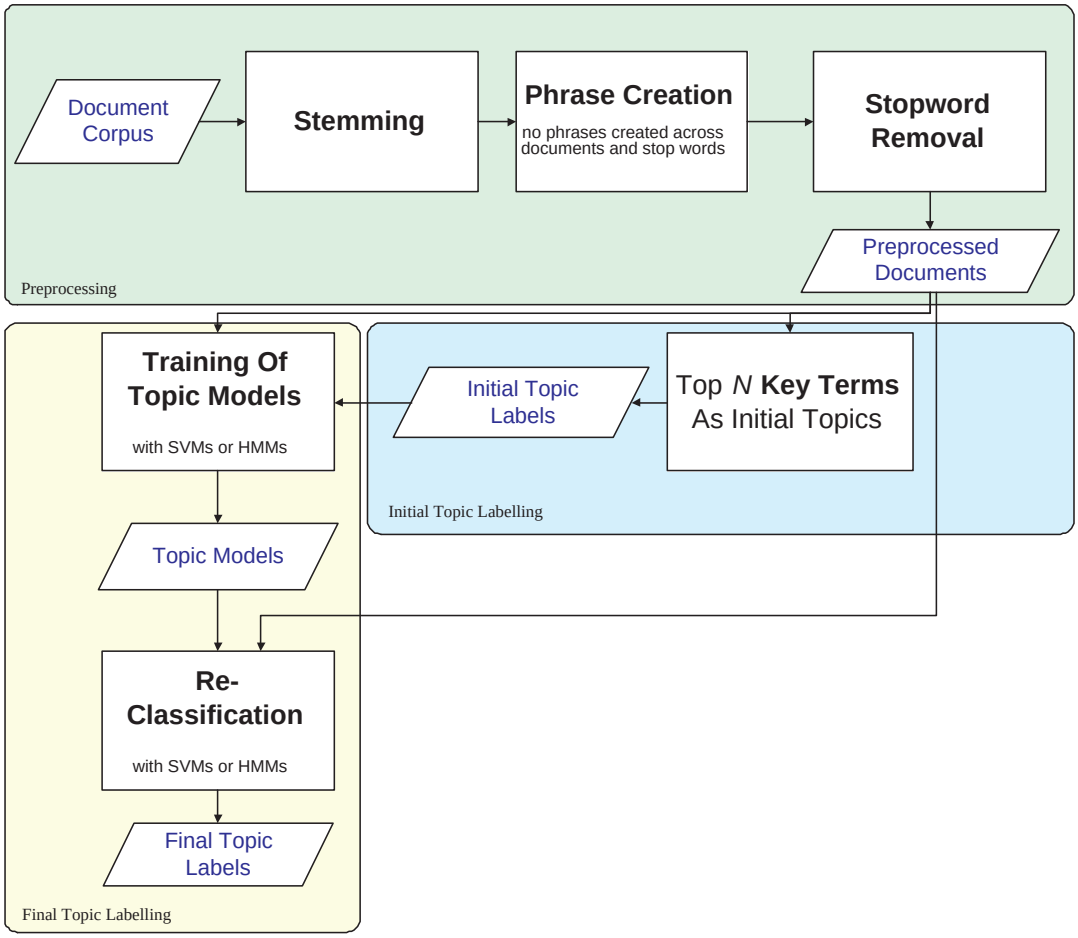


Figure 10.1: Overview of Unsupervised Topic Discovery.

tends to remove suffixes from the words more rigidly that the stemmer from University of Dortmund [61], i.e. the stemmed words are somewhat shorter.

Phrase creation

The next preprocessing step is to find phrases in the text and combine the words that make up the phrases. A phrase consists of more than one word and can provide a better description than a single word. For example, the phrase "United Nations" is more descriptive than the words "United" and "Nations" alone. The goal of the phrase creation is to find meaningful phrases, and then to combine them. After this step, the text "chancellor schröder"[1] will become "chancellor_schröder".

[1]The output of the automatic speech recogniser consists of lower case words only.

Only a sequence of words that substantially decrease the description length (DL) of the entire corpus is transformed into a phrase. The description length is the number of bits that is needed to represent the whole corpus in a computer memory. It consists of two parts: the coded text, where each word is represented by its numerical integer index, and a code table (or vocabulary) that contains the string of each word and the corresponding index. The initial description length is therefore (assuming 8 bits per byte, and $l(W_i)$ the number of characters in word W_i)

$$DL = \text{size of vocabulary table} + \text{size of coded text} \tag{10.1}$$

$$= \sum_{W_i \in \text{corpus}} 8 \cdot l(W_i) - \sum_{W_i \in \text{corpus}} c(W_i) \log_2 p(W_i). \tag{10.2}$$

$c(W_i)$ is the number of times W_i appears in the corpus. Note that both the lower-case w_i and the upper-case W_i refer to a word, but the meaning of the index is different. w_i means the i-th word in a text or in a sequence, whereas W_i stands for the i-th word in the vocabulary.

When two terms are merged into a phrase, the phrase has to be included into the vocabulary table. The new entry in the vocabulary does not contain the string representation of the whole phrase, but just two indices which refer to the words in the phrase. Assuming 32-bit integer indices, the entry occupies 64 bit.

Phrases can also contain more than two words, but in one single phrase combination step, only two terms (words or an already existing phrase) are combined. To get a three-word phrase, first two words are combined into a phrase (e.g. "gerhard schröder" → "gerhard_schröder"), and in the next step, the third word is merged with the initial phrase: "chancellor gerhard_schröder" → "chancellor_gerhard_schröder".

To get the change in description length, $\Delta DL(w_1 w_2)$, when two terms $w_1 w_2$ are merged into one phrase, one has to add the change caused by removal of w_1 and of w_2. The change in bits caused by removal of w_1 is symbolised by T_1, the change in bits caused by removal of w_2 is symbolised by T_2, where $T_1, T_2 < 0$. Note that not all w_1 disappear, but only those which are followed by w_2. The easiest way to calculate T_1 is to count the bit change for *all* w_1, and then to subtract the bits for which w_1 is *not* followed by w_2:

$$\begin{aligned} T_1 =& c(w_1) \log_2 p(w_1) \\ & - [c(w_1) - c(w_1 w_2)] \log_2 \left(\frac{c(w_1) - c(w_1 w_2)}{N - c(w_1 w_2)} \right). \end{aligned} \tag{10.3}$$

In (10.3), the number of times w_1 is not followed by w_2, $c(w_1) - c(w_1 w_2)$, is divided by the total number of terms in the *new* document corpus (after phrase merging). A descriptive explanation of T_1 is that *all* occurrences of w_1 are removed from the old text (before phrase

merging), and that only those w_1 are put back into the new text that are not followed by w_2. T_2 can be calculated in a similar way.

The number of bits for the new term w_1w_2, the change in bits for all other terms, and the 64 bits for the new entry in the vocabulary also contribute to $\Delta DL(w_1w_2)$. Empirical observations have shown that the change in bits for all other terms contributes least to $\Delta DL(w_1w_2)$ (ignoring the 64 bit constant). The exact formula for $\Delta DL(w_1w_2)$ can be found in eq. (4-2) in [82].

The algorithm for creating the phrases is very similar to Sista's approach [82, p. 91]:

Algorithm 10.1 Algorithm for creation of phrases.

1. Replace each term with its index in the vocabulary table
2. Compute the word counts $c(w)$, phrase counts $c(w_1w_2)$ and the total number of terms in the corpus N
3. Calculate $\Delta DL(w_1w_2)$ for each phrase that appears at least three times. Do not create phrases across document boundaries, and if one term is a stop word. Word consisting of one or two characters are also ignored for phrase creation.
4. If $\Delta DL(w_1w_2) <$ threshold, replace the separate terms w_1w_2 with the phrase w_1w_2. Update the vocabulary if a phrase was created.
5. Repeat steps 2 to 4 until no new phrases are created.

A good value for the $\Delta DL(w_1w_2)$ threshold is -10. Ignoring phrases that appear only once or twice significantly speeds up the algorithm. After phrase creation, 154 manually selected stop words are removed from the documents.

10.3 Key term identification

The next step of the UTD algorithm is to identify so-called key terms, i.e. the most relevant terms of a document. For each story, the corresponding key terms are used as the initial topic annotation for that story; the key terms are used as topic labels themselves. However, this topic annotation is only able to find topic labels if the label appears as a key-term in the story. But a story may also have a meaningful topic, even if the corresponding word does not appear in the text. To overcome this drawback, a classifier is trained with the initial topic labels as reference labelling. The documents are then re-labelled by classifying them with the trained models. This approach allows the detection of topics even if the corresponding term does not appear. One way to get the key terms of a document is to assign a TF-IDF weight (see section 8.3.2 on page 102) to every term in a document. TF-IDF is a term importance or relevance indicator; it is a measure for the relatedness of the term to the contents of a document.

For every term of a document, its TF-IDF weight is computed according to eq. (8.2). For each document, the $I = 5$ terms with the highest weight are chosen as initial topic labels for the document. These top-ranking terms are called *key terms*. It is not always possible to select exactly I key terms, since in several documents there are terms that have equal TF-IDF weights. If the $(I + 1)$-th term has got the same weight as the I-th term, it also becomes a key term. And so do the following terms with the same weight. For example, if the 4-th to 7-th term (ranked according to their TF-IDF) have got the same TF-IDF weight, then the top 7 terms become the initial topic labels[2].

If a top-ranking term appears only in one or two documents, the corresponding topic will have only few training documents in the subsequent re-classification step. Consequently, terms which appear only one to three times in the document collection are skipped and do not become initial topic labels.

10.4 Document re-classification

According to Sista's observation, already the key terms, and therefore the initial topic weights describe well a document. 92 % of the top-ranked key terms are correct, while 81 % of the 4-th ranked and 68 % of the 5-th ranked key terms are correct [82, Table 5-8]. But the initial topic labels are restricted to those terms that appear in the corresponding document. A topic label that would well describe a specific document will never be chosen unless it appears as a term in this document. To overcome this drawback, Sista suggests to train Hidden Markov topic models using the initial topic labels and to re-classify all documents using the trained models. This approach is described in the following section. An alternative to Sista's Hidden Markov Model-based classifier is a SVM classifier (Chapter 3 on page 14), which will be treated afterwards.

10.4.1 HMM topic classification

In contrast to the HMM topic classifiers presented in chapter 7 on page 80, the HMM classifier used for the UTD task is able to explicitly deal with multiple topic labels per document. It is based on the assumption that every word in a story is generated by a different topic (see Figure 10.2). When a story starts, a set of topics is chosen with a probability of $P(Set)$. This set remains fixed for the story. Among the M topics in the set (which always include the background model *General Language*), one topic model (one single HMM state) T_i is chosen

[2] Sista's algorithm always chooses the top $I = 5$ terms. Maybe he did not observe terms with identical weights, due to a much larger document collection (45,000 compared to roughly 500 used here).

with $P(T_i|Set)$. A word is then emitted with a probability of $P(W_n|T_i)$. Before the next word is emitted, again a topic has to be chosen. Thus, every word can originate from an individual topic model, independent of other words. The following constraints hold:

$$\sum_{T_i \in Set} P(T_i|Set) = 1 \qquad \text{and} \tag{10.4}$$

$$\sum_{W_n \in T_i} P(W_n|T_i) = 1. \tag{10.5}$$

The General Language model collects all words that are not specific of the other topics in the set; a General Language topic label is added to the initial labels of *each* story.

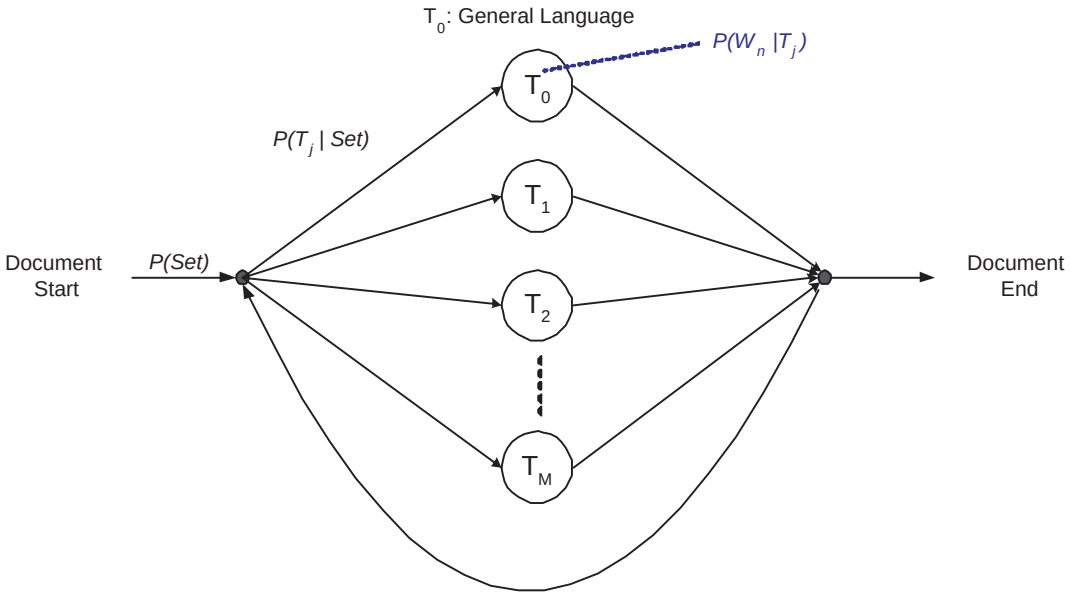


Figure 10.2: HMM Topic Set Model for UTD.

Estimation of model parameters

All documents, which are annotated with initial topic labels, form the training set to estimate the HMM parameters. The parameters are chosen in such a way that they maximise the likelihood $P(d|Set) = (\prod_{i=1}^{n} P(d_i|Set))^{1/n}$, where a d_i is one of the n documents that have the Set of topics as reference labels.

From the training data, it is known which words are emitted, but it is not known by which HMM state (by which topic). The solution of the maximum likelihood (ML) optimisation problem is the EM algorithm (see Section 2.2). The hidden data is the information about which topic model (HMM state) has produced the observed words of a document; it is only

known (from the set of topics assigned to the training documents) which topic set has produced the document.

Each topic Set can be considered a Hidden Markov Model with the parameters

- $P(T_i|Set)$ and
- $P(W_n|T_i)$.

Unlike the usual HMM parameter estimation (chapter 2), where the transition and emission probabilities are estimated separately for each HMM, here the topic distributions $P(W_n|T_i)$ are calculated globally for all HMMs (hence, $P(W_n|T_i)$, and not $P(W_n|T_i, Set)$). Similarly, a global transition probability $P(T_j|j \in Set)$ is estimated (denoting the global (average) probability of a topic given that it occurs in an arbitrary Set). The HMM-specific $P(T_j|Set)$ are derived by scaling $P(T_j|j \in Set)$ to sum to 1 for the Set of the HMM.

The EM parameter estimation is done according to Algorithm 10.2. Several remarks about it have do be made. The denominator in (10.7) equals the probability that the word W_t has been produced by the Set of topics:

$$P(W_t|Set) = \sum_{T_i \in Set} P(T_i|Set)P(W_t|T_i). \tag{10.6}$$

The ratio in the same equation tells to what degree (between 0 and 1) the topic T_j (among the topics in the Set) contributes to the word W_t. Since this contribution is summed over all stories, $C(W_t, T_j)$ can be paraphrased as

$$\begin{aligned} C(W_t, T_j) =& (\text{average, relative contribution of } T_j \text{ to } W_t) \cdot \\ & (\text{\# of stories}). \end{aligned}$$

The denominator in the calculation of the emission probabilities (10.8) can be paraphrased as

$$\begin{aligned} & (\text{\# of stories}) \cdot (\text{size of vocabulary of } T_j) \cdot \\ & (\text{relative contribution of } T_j, \text{ averaged over all words in the vocabulary of } T_j). \end{aligned}$$

The vocabulary of a topic T_j consists of all words that appear in stories labelled with T_j. Sista uses a different denominator for (10.8) (namely, the total word count in all stories labelled with topic T_j), which makes $P(T_j|j \in Set)$ not being a probability since it might get larger than 1. Even more, there is no "word count" implied in the above equations (just a vocabulary size, which is something different). Therefore, we believe his denominator is not justified.

To understand the need for a bias b_j in (10.9), let us consider the following, simplified example:

Algorithm 10.2 EM-Estimation of parameters of HMM topic models used for UTD.

1. **Initialisation**. Initialise $P(W_t|T_j)$ by counting number of occurrences of each word in those stories that are labelled with T_j, then normalise to form probabilities ($\sum_{W_n \in T_j} P(W_n|T_j) = 1$).
   Set $P(T_j|Set)$ as the ratio of number of stories labelled with T_j to the total number of stories, and normalise for each story.
2. **Expectation Step**. Estimate the counts of all (word, topic) pairs with

$$C(W_t, T_j) = \sum_{\forall stories} \frac{P(T_j|Set)P(W_t|T_j)}{\sum\limits_{T_i \in Set} P(T_i|Set)P(W_t|T_i)}. \tag{10.7}$$

3. **Maximisation Step**. The HMM parameters, i.e. the transition and emission probabilities, are re-estimated. To get the emission probabilities, the counts $C(W_t, T_j)$ have to be normalised:

$$P(W_t|T_j) = \frac{C(W_t, T_j)}{\sum\limits_{\substack{\forall \text{words } W_i \text{ that appear} \\ \text{in the stories labelled with } T_j}} C(W_i, T_j)}. \tag{10.8}$$

   The transition probabilities tell how much a topic contributes to the creation of a story. They are estimated using

$$P(T_j|j \in Set) = b_j \cdot \frac{\sum\limits_{\substack{\forall \text{words } W_i \text{ that appear} \\ \text{in the stories labelled with } T_j}} C(W_i, T_j)}{(\text{\# of stories}) \cdot (\text{size of vocabulary of } T_j)}. \tag{10.9}$$

   b_j is the bias of topic T_j according to (10.12).
4. Steps 2 and 3 (Expectation and Maximisation) are repeated for several iterations.
5. Un-biased transition probabilities are computed using (10.9) and a bias of $b_j = 1 \, \forall j$.

Assume topic T_j emits only two different words, but these words are typical of T_j, so that rarely any other topic emits them. The numerator of eq. (10.9) might then be

$$\sum C(W_i, T_j) = \text{\# of stories} \cdot (0.95 + 0.91), \tag{10.10}$$

and this would have to be divided by

$$(\text{\# of stories}) \cdot 2$$

to get $P(T_j|j \in Set)$ (ignoring the bias coefficient):

$$P(T_j|j \in Set) = 0.93. \tag{10.11}$$

This means that although T_j has only two significant words, a transition value of 0.93 would wrongly suggest that it emits most of the words. Hence, the bias b_j is included that is large for large topics, and small for small topics:

$$b_j = \frac{\sum\limits_{\substack{\forall \text{words } W_i \text{ that appear} \\ \text{in the stories labelled with } T_j}} C(W_i, T_j)}{\sum\limits_{\forall T_i} \sum\limits_{\substack{\forall \text{words } W_i \text{ that appear} \\ \text{in the stories labelled with } T_j}} C(W_i, T_j)}. \tag{10.12}$$

Restricting topic models to support words

Sista only keeps those words in the topic distributions $P(W|T)$ for which $\frac{P(W|T)}{P(W)} \geq 1$.[3] They are called *support words*. Unfortunately it does not become clear how exactly the non-support words are removed. One way would be to set their likelihoods $P(W|T)$ to 0, before or after the EM estimation (Algorithm 10.2). The more correct way is to do it after EM estimation, but it saves run time to do it before.

Due to the zero frequency problem (see section 7.2.1 on page 82), the likelihoods $P(W|T)$ have to be discounted or smoothed before calculating the posteriors $P(T|d)$. For the presented UTD experiments of this thesis, all $P(W|T)$ are interpolated with the unconditional word probability:

$$P_{\text{new}}(W_i|T_j) = 0.75 \cdot P(W_i|T_j) + 0.25 \cdot P(W_i). \tag{10.13}$$

[3] In his thesis [82], he writes that all words with $\frac{P(W|T)}{P(W)} > 1$ are *discarded*, but this is obviously a misprint, since it would mean that those words are removed that are specifically significant for a topic.

The interpolation weights are set according to Schwartz et al. [79], who also describe the HMM topic classifier used by Sista [83].

For our experiments, it turned out that the restriction of the topic models $P(W|T)$ to contain only support words is only a matter of reduction of run time. The final topic labels are almost identical, whether words are removed from the topic models or not.

Re-classification

Each document is classified based on the trained topic HMMs, i.e. it is assigned a set of topics. Since the initial topic labelling step can also be seen as a kind of classification, this HMM-based final classification is referred to as re-classification.

Given the training assumptions and the document generation process in Figures 10.2 and 10.3, each possible Set of topics should be considered, and the Set with the highest posterior probability, $P(Set|d)$ should be assigned as the classification result. However, there are hundreds or thousands of topics, therefore it is computationally infeasible to compute the posterior probability for all combinations of topics.

Fortunately, it is sufficient for UTD [82] to calculate only the (log) posterior probability of a single topic, $P(T_j|d)$, and then to assign the N best topics as topic labels:

$$\log P(T_j|d) = \log P(T_j) + \sum_{\forall W_t \in d} \phi \left(\log \frac{P(T_j|j \in Set)^{\beta} P(W_t|T_j)}{P(W_t)} \right). \tag{10.14}$$

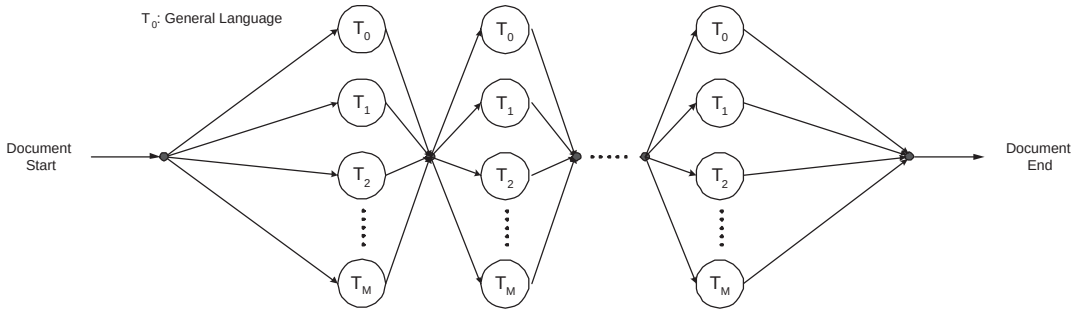


Figure 10.3: Assumed document generation for HMM training (compare Figure 10.2).

This formula treats each topic individually, and implies that each word in the test document has been generated by the same, unique topic T_j (as depicted in Figure 10.4). This is a clear violation of the assumptions made for training, namely that each word is generated by another topic state (as shown in Figure 10.3). Most words of a document are attracted by the General

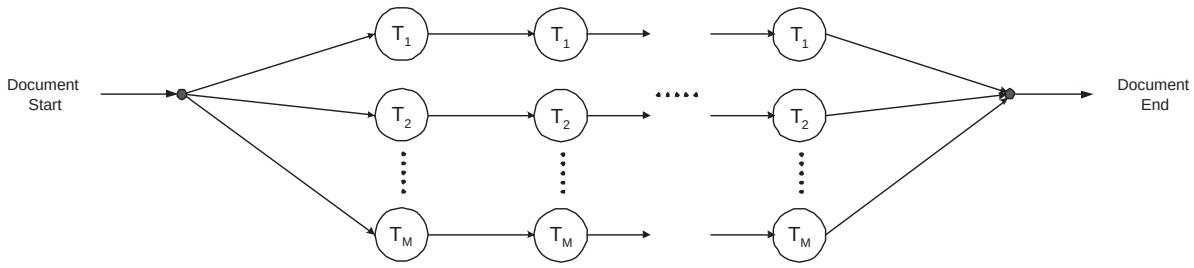


Figure 10.4: Document generation assumption made by eq. (10.14).

Language state, which is not reflected in (10.14). To account for words that are *not* created by T_j, eq. (10.14), incorporates a filtering function ϕ with

$$\phi(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad . \tag{10.15}$$

Although the usage of ϕ is theoretically justified, it implies significant problems that were overlooked by Sista. Due to ϕ, only zero or positive log probabilities are added. The right hand sum term is therefore necessarily positive (or zero), and the term $\log P(T_j)$ is usually not negative enough to make the log posterior probability be negative. Talking about probabilities, and not log probabilities, this means that the posterior probability can easily become greater than 1! This contradicts one of the fundamental properties of probabilities, and it is questionable whether the resulting probability is still some kind of "normalised" and can directly be used for topic ranking. Compare, for example, the likelihood used in conjunction with confidence measures (Section 7.2.3). It is not normalised, and can therefore not be used as a ranking measure. Nevertheless, Sista reports very good UTD results using his approach [82].

Even more, we have observed that due to ϕ, for many, if not most topics and documents, the sum term (last term in eq. (10.14)) is zero. The posterior probability of a topic given the document then becomes the a-priori probability of the topic: $P(T_j|d) = P(T_j)$. The $P(T_j|d)$ values, and hence the topic ranking (which is done according to $P(T_j|d)$) then becomes completely useless. Therefore, a filtering of the probability ratio was not performed for our experiments. The exponent β is introduced due to the wrong independence assumption and is set to $\beta = 0.35$ according to [79].

For each story d, the posterior probability of each topic is calculated according to (10.14), and the best topics are chosen as the final topic labels of d.

10.4.2 SVM topic classification

An alternative to the HMM topic classification as described in the previous section is the classification with Support Vector Machines (SVMs, see Chapter 3 on page 14). As the UTD task incorporates more than two labels, but SVMs can only deal with two classes, the multi-class problems have to be broken down into binary problems. The SVMs used for the topic classification in Chapter 8 on page 99 train a separate model for each class pair (one-against-one, see section 3.5 on page 26). This approach to multi-class categorisation is not feasible for the UTD task, since each document has got more than one training topic. When training the binary model for (T_i, T_j), all stories would have to be excluded that are labelled with *both* T_i and T_j. This would unnecessarily reduce the amount of training stories, and maybe even prohibit reasonable model training for some topic pairs, since there are not enough training documents.

However, with an one-against-all approach (Section 3.5), all training stories can be used (see also [43]). The model for topic T_i is trained with all stories that are labelled T_i as the positive class, and as the negative class all stories that do not have T_i among their topic labels. Non-probabilistic SVMs output the class with the highest decision value as the class prediction, but they do not provide a ranked list of best topics. One could think of ranking the classes according to the decision value that was achieved with the corresponding binary SVM model, but as was pointed out in section 3.5.1 on page 27, the decision values are not comparable across binary models.

A better approach is to estimate posterior probabilities $P(T_j|d)$ with probabilistic SVMs (pSVMs, Section 3.4), and then to rank the topics according to the posterior probability. However, pSVMs deliver only an (estimate of a) binary a-posteriori class probability $P(T_i|d, \text{model } T_i)$ for each model. An estimate for $P(T_j|d)$ can be easily derived by normalising the binary posteriors to form a probability: Divide all $P(T_i|d, \text{model } T_i)$ by the same constant $c = \sum_{\forall T_i} P(T_i|d, \text{model } T_i)$ so that

$$\frac{1}{c}\sum_{\forall T_i} P(T_i|d, \text{model } T_i) = 1, \qquad \text{then} \tag{10.16}$$

$$P(T_j|d) \leftarrow \frac{1}{c} P(T_i|d, \text{model } T_i). \tag{10.17}$$

The final topics for each document d are the best topics ranked according to their $P(T_j|d)$.

For the topic re-classification with SVMs, each document is represented in the vector space model (see section 8.3.1 on page 102 for details). From the word representation of a document, a (sub-) feature vector is created; the same is done for the character 3-gram representation.

Both sub-features are concatenated to get the final feature vector. The word sub-vector is weighted with $w_{\text{word}} = 1.5$, since experiments in chapter 8 indicate that this choice often leads to good results. The trade-off parameter C is set to 1.3, just as for the experiments presented in chapter 8.

From a theoretical point of view, the SVM approach allows a cleaner representation than the HMM approach: there is no need for a filtering function ϕ. As SVMs are discriminative classifiers, and in contrast to generative HMMs that estimate probability density functions (pdf), they need fewer stories per class to train a model. However, the estimated pdf allows insight into the degree of contribution of certain words to a specific topic. This information cannot be easily taken from the SVM models. The experiments in the following section will compare the performance of both approaches.

10.5 Experiments

For the texts used for the UTD experiments, it was observed that the two values (1) size of the vocabulary table and (2) size of the coded text contribute almost equally to the DL of the text corpus.

10.5.1 Data sets

Tow different data sets were examined:

- **ASR set**. The first set consists of the automatic transcription of 443 TV broadcast news from two weeks in October 2001. This is the same data set that has been used as test set for the SVM topic classification experiments This is the same data set that was used as test set for the SVM topic classification experiments (chapter 8).
- **DLF set**. The second set was extracted from the mail newsletter sent by the *DLF (Deutschlandfunk)* radio channel. It contains the manuscripts of the bi-hourly five-minute radio news, is usually error free, and is segmented according to topic boundaries. 591 stories were extracted from four broadcast manuscripts per day, covering the first two weeks in December 2004. After stop word removal, 26,511 words are left over.

Both sets cover two weeks, and have roughly the same number of stories. The second set, however, consists of (nearly) error free text, stories are much shorter (less than one minute), and variance of story length is smaller than the ASR set. The subsequent experiments were

usually conducted with both data sets. If no data set is explicitly mentioned, the statements and conclusions are valid for both sets.

Neither data set contains a reference topic labelling that can be used to automatically evaluate the UTD results. While the ASR set was labelled with topics useful for media monitoring (see the previous chapters), only few stories contain labels (except for the *Off-topic* label), and only one label per story. Even more, these labels do not necessarily tell much about the content of the story, but more about who might be interested in the story. Therefore, this labelling cannot be used for UTD.

Manual evaluation is very time consuming. Several people are needed to check every topic created by the UTD algorithms, and if several runs with differing parameters shall be evaluated, this approach becomes infeasible. Sista [82] has chosen 100 stories to be evaluated, and let humans judge whether a topic is true or has to be discarded. For every topic rank, the precision of the UTD topic annotation is calculated separately. The recall (fraction of true topics that were found by the system) is not computed, since this would need manual annotation of the stories, and it is not clear at which granularity the true topics should be chosen (recall that reference annotators are not restricted to choose from a fixed number of topics, since there is no pre-defined topic list).

All judgements made in this chapter about whether a topic is good or not do not use a defined evaluation measure, but are based solely upon manual inspection of randomly selected stories.

10.5.2 Stemming

It has not been finally decided in the text classification community whether stemming is useful or not (compare Section 8.2). For text classification with SVMs, we observed that stemming does not improve results. However, stemming of German with its abundant grammatical morphemes should be beneficial for the UTD task. Consider, for example, a word that appears in the document collection a few times in its singular form, and a few times in its plural form. The individual singular and plural forms might get a low TF-IDF weight, whereas the stemmed form (singular and plural reduced to a common stem) might get a high weight. Moreover, the individual forms may be excluded from the initial topic list, since they appear less often than the minimum threshold for key term selection. The stemmed form then would be selected since it appears often enough. These examples clarify why in the key term selection phase, stemming might be useful.

To examine this claim, UTD was performed with and without stemming (using the SVM classifier and the DLF data set). As far as key term selection is concerned, there is a tendency that stemming leads to better initial topics. However, this depends on the story. For some stories,

the initial topics do not differ much when created based on stemmed or non-stemmed texts. Some stories even have better initial topics with non-stemmed texts. As a consequence, words were always stemmed for the subsequent experiments.

10.5.3 Phrase creation

One free parameter that has to be chosen a-priori is the threshold of $\Delta DL(w_1 w_2)$ which determines how many term couples will be merged into a phrase. Unfortunately, this threshold has to be set dependent on the document collection. For our collections, a threshold of -10 bits seems appropriate for the data sets considered, where the lowest $\Delta DL(w_1 w_2)$ is -541 bits for the DLF set (phrase *neblig_trüb*, foggy-cloudy) and -509 for the ASR set (phrase *new_york*).

With a collection size of 45,000 documents, there are five-digit $\Delta DL(w_1 w_2)$ [82]. One option would be to set the threshold relative to the description length of the whole corpus, but as our observations have shown, this approach is not feasible.

10.5.4 Key term selection

The terms with the highest TF-IDF weight, are usually able to give good hints about the contents of a story, and allow a good description.

However, not all of the best-weighted terms are chosen as initial topic labels since they appear only a few times throughout the document collection ($DF < 4$, see Section 10.3). It turns out that especially these infrequent terms allow the user to get a good insight into the document's contents.

A typical property of key terms is that they do not find a broad topic label for a story, but each term picks one out of several aspects; they are quite focused. Only the conjunction of key terms will indicate the contents of a story. One good example is the weather forecast. A user would like to get a topic label like *forecast* or *weather_forecast*, but this label does not appear (nor does it as a final topic after re-classification). Instead, the top key terms are only related to weather, like *wolk[en], grad, sonn[e], schau[er], süd[en]* (clouds, degree, sun, rain, south).

The key terms, taken on their own without additional information, do not provide a summary, but they give a hint to the user. It is often helpful to have a certain knowledge about e.g. politics and the themes that are currently being discussed; given this knowledge, it is possible to interpret the key terms so that they in fact deliver a summary. Among the terms with a high TF-IDF, those that appear in more than a few documents are (not surprisingly) more general than the terms which appear only once or twice in the collection.

It is not possible to set a fixed threshold on the number of top key terms that are to be chosen as topic labels. This is especially true for stories that have a lot of high-ranking terms with low document frequencies (DF): A key term that appears only in two or three documents can tell very much about their contents, and if there are 10 such key terms, all should be kept as topics. One reason is that those infrequent, top key terms often have exactly the same TF-IDF, since they appear equally often (both in the document (TF term), and in the document collection (IDF term). Then, a fixed threshold, which keeps only, say, three terms although six have the same TF-IDF weight, does not make sense.

Of course, not all topic key terms are valid and indicate a story's contents. But wrong key terms disrupt understanding of a story less than one would expect. As far as ASR texts are concerned, due to errors in speech recognition, it occurs that deficiently transcribed words are selected as keywords (e.g. *AWACS* $\rightarrow$ *airbags*). Therefore, the list of key terms is worse for the ASR set than for the DLF set.

Although the main conclusion of this section is that also terms with a high TF-IDF and a DF of 2 or 3 should be selected as initial topic labels, the following re-classification experiments are based on key terms with $DF \geq 4$. Inclusion of the remaining key terms would lead to training of topic models based on 2-3 stories, which is a bad idea especially for estimation of HMM emission probabilities.

For the DLF set, 1158 key terms (with $DL \geq 4$) are selected as initial topics. The corresponding number is higher for the ASR set (1261 initial topics), although it contains slightly less documents.

10.5.5 Re-classification with HMMs and SVMs

Most of the final topics obtained from the re-classification are completely different from the initial topics, and they actually provide a much worse representation of the document. This is true for both the SVM and the HMM classifier. There may be several (around 10) *key terms* that are good, but any *final topic* that is at rank 6 or lower is usually wrong. The precision of the higher-ranking final topics is also significantly worse than the precision of the key terms. And more often than not, the final topic is wrong.

For example, one story from the ASR set about the unemployment rate in the month of September has got the key terms *september, arbeitslos, beschäftigung, tausend, wirtschaft*[4] (September, unemployed, employment, thousand, economy). The final topic labels (from the SVM classifier) are: *festgestellt, wolfsburg, infektion, mannheim, endspiel, beitrag* (asserted, [city

[4]The keyterms are stated here in their complete form, but the UTD will output them in their stemmed version.

Table 10.1: Sample ASR output text, initial topics, and the top six final topics as found by the HMM and the SVM re-classifier.

ASR output (not stemmed)
unter den laden hatte die amerikaner geplant niemand könne die jetzt mehr mitgefühl die angst vor neuen terroranschlägen die den usa noch einmal gewachsen aber dennoch die ganz grosse mehrheit der amerikaner unterstützt die militäraktion vierundneunzig prozent befürworten die angriffe auch auf ganz da wir waren der neue und die herausforderungen sind komplett in weiten haus wohl kaum mitleid direktor der behörde für heimat verteidigung vereidigt auch heran will den amerika für verhindern nach mitteln zentraler punkt der erde sein oder fremder kalt ihr werden unterland verteidigen ohne die einmaligen freiheiten und weil die beste verteidigung ist eine globale unfällen wie perfekt wo immer der terror auch auftauchen war der euro fällt jede der vier uhr erwartet weil dich herr in der traumwelt würden zukunft und dort für ämter koordinieren die sich bereit erklärt mit dem die lintfort beschäftigen wird neun hektar der wir müssen vor allem möglichst viele geheimdienst informationen zusammentragen im kampf gegen den terror ist werden macht im wendland jähriger wenn mittlerweile die sicherheitsvorkehrungen verschärft worden zahlreiche strassen die hier vor dem aussenministerium in washington wurden gesperrt viele amerikaner will nach den angriffen auf afghanistan verunsichert er werde dealer die die terroristen verfolgen oder nicht eine vielfach klar wir werden uns weiter verfolgen wir hatten werden paar gerät das ganze kleine lang andauernder aufgabe die terror treffen von weiteren anschlägen abteil äh äh äh ich freue will er wir amerikanern nehmen die verstärkten sicherheitsvorkehrungen in sport stadien oder flughäfen ohne murren hin wenn jeder hier wurde dass die militäraktionen der amerikaner neue anschläge in den usa auslösen könnten
Initial Topics (key terms)
verteid, sicherheitsvorkehr, verfolg, amerikan, militaraktion
Final Topics from HMM re-classification
palastinens, prozent, grad, wolk,sech, taliban
Final Topics from SVM re-classification
amerikan, militaraktion, verfolg, sicherheitsvorkehr, mannheim, sparkur

of] wolfsburg, infection, [city of] mannheim, final match, contribution). None of the initial topics (key terms) is any more present in the final topics. Actually, the key terms describe the story much better than the final topics. Interestingly, the city name Mannheim appears among the top 10 final topics in almost all stories. Obviously, its SVM model is not very good. Another example is given in Table 10.1.

The final topics from the HMM classifier are even worse than the SVM topics. Comparing the topics of all documents, the topic list does not change very much, i.e. more or less the same topics appear in many documents.

One can attribute the bad performance of both classifiers to the small number of stories used to train the final topic models, which in turn results from the small document collection used. For example, for the DLF set and HMM re-classifier, the final topics *cdu* and *grad* are assigned to (almost) every story. The model for *cdu* is trained with 27 sample stories, the model for *grad* is trained on 20 stories. In contrast, the initial topic *beauftragt* does not appear as final topic; its model is trained on only 2 stories. This indicates that there is some relationship between training size and appearance as final topic. This aspect demands further investigation in order to draw well-founded conclusions.

With a document collection that is two orders of magnitude larger (45,000 documents covering 12 months, or 125 (!) documents per day), Sista [82] reports that 96 % of the first-ranked final topics are correct (as judged subjectively by a human evaluator). Re-classification adds only a small fraction of topics (about 9 %), compared to the initial topics. The precision of the initial topics is lower, although it is not always statistically significantly lower. 92 % of the initial first rank topics are correct (as opposed to 96 % final topics); for the fourth rank, the precision is 81 % versus 82 %.

Combining both results

- For a small document collection, re-classification significantly worsens results
- For a large collection, re-classification improves results, but not necessarily statistically significantly

one can conclude that unless the document corpus to be annotated is very, very large and there is enough (run-)time available, re-classification should not be performed.

As we have observed that the results with SVM re-classification is better than with HMM re-classification, it should be nevertheless worthwhile to check the performance of SVMs on a very large broadcast news corpus.

10.6 Conclusion

This chapter introduced the task of Unsupervised Topic Discovery. Its key characteristics are that topics are assigned to the text documents of a collection in an unsupervised way, and that the list of topics is not pre-defined, but is derived from the collection. There is no need for human interaction or topic labelling of training texts. The UTD approach by Sista was presented together with modifications which were justified. As an alternative to his HMM re-classification, an SVM classifier was suggested and investigated. Experimental results based on relatively small corpora of both ASR and clean broadcast news texts were discussed.

The main conclusions to be drawn from this chapter are:

- The choice of the threshold value for phrase creation is not obvious and depends on the size of the document collection.
- The terms with highest TF-IDF weights (key terms) should be used as final topic labels, and only those with $DF = 1$ should be discarded (but not those with any higher DF).
- The restriction of HMM topic models to contain only support words (as suggested by Sista) has very little effect on final topic classification. This might also partly be due to the fact that the HMM topic models used for the experiments presented in this thesis were quite poor.
- As the key terms can only be drawn from one document, speech recognition errors may have quite a big impact on the quality of the topic labels.
- For small corpora, the re-classification step delivers bad results and should therefore be omitted.

Chapter 11

Conclusion and outlook

The subject of this thesis is the development of a demonstrator for automatic media monitoring that automatically scans TV broadcast news for specific topics. It is the first media monitoring system for German news. Its goal is to filter out relevant stories from a stream of broadcast news.

In the course of development, new techniques were conceived and existing techniques were improved to create the modules of the demonstrator. Many experiments were conducted to measure the performance of the various approaches.

In addition to the media monitoring task with its pre-defined list of topics to be identified, the problem of Unsupervised Topic Discovery was addressed and investigated. It is nearly unexplored, but is very attractive as it aims at identifying topics (or key-words) which do not have to be defined (by means of training data) in advance. In contrast to the media monitoring approach, which only filters out *certain* stories, the UTD approach assigns key-words to *all* stories.

Automatic speech recognition The automatic speech recognition of broadcast news (Chapter 5 on page 47) demands special approaches. Recognisers not specifically designed for this task perform poorly. It turns out that by using our language model that is based on newspaper texts instead of a general language model, the word error rate (WER) of the speech recogniser is reduced by 43 % relative. The interpolation of the newspaper language model with a language model based on the manual transcription of broadcast news causes a further reduction of the WER by 12 % relative. Another effective method is to use monophone and triphone acoustic models trained on broadcast news. Compared to triphone acoustic models trained on spontaneous speech and read sentences, our experiments reveal that an improvement of one third (monophone broadcast news models) and of one quarter (triphone broadcast news models) is achieved. Other measures, for example improvements of the dictionary or gender-

dependent acoustic models, also lead to the reduction of transcription errors, but to a minor degree. The representation of the dictionary as a tree as well as the caching of the language model in a FIFO are crucial to make the recogniser efficient in terms of run-time and memory requirements.

The quality of the generated transcription depends much on the presence of background noise. Further improvements should therefore concentrate on delivering better transcriptions when there is background noise.

Topic segmentation A novel, visual approach to topic segmentation of TV broadcast news (Chapter 6 on page 55) uses Hidden Markov Models (HMMs) to represent content classes (e.g. Newscaster, Report, Interview) and edit effects (e.g. Cut, Dissolve, Wipe). These models are combined into a lattice (hierarchical model) that reflects how a news show is built up from content classes and edit effects. For topic segmentation, the path through the lattice is found that has most probably created the observed feature sequence. This step creates a series of content classes and edit effects. Rules that define the beginning of a new topic are subsequently applied.

Our experiments show that more complex lattices capturing more variants of paths do not necessarily yield better results. Lattices that capture the structure of topics in addition to the news show structure yield worse segmentation performance. It turns out that it is sufficient to extract features at a reduced rate of 12.5 frames per second, since segmentation results do not necessarily improve with 25 frames per second. The segmentation module achieves recognition rates of up to 88 % precision and 82 % recall. However, its performance varies with station and programme. One drawback of the presented segmentation approach is that for every station, possibly even for every news programme, news show lattices have to be defined. In the future, besides vision, other modalities like automatic transcription should be exploited.

Topic classification For topic classification based on automatic transcription, three methods were investigated in our work. The well-known Naive Bayes approach serves as a baseline system (Section 7.2 on page 81). A novel method uses a Hidden Markov Model classifier and quantised feature vectors (Section 7.3 on page 84). The quantisation prototypes are created with a Neural Network trained according to the Maximum Mutual Information principle. The novel approach is superior to the Naive Bayes approach when it has a limited choice from only few training topics. Thus, it is suitable as a topic classifier for a media monitoring system if only a rough discrimination between topics (e.g. politics, economy, sports, ...) is needed.

The third method, Support Vector Machines (Chapter 8 on page 99), is frequently used for text categorisation. However, this thesis investigates for the first time probabilistic and non-

probabilistic couplers for topic classification. Couplers are used to combine the results of two-class Support Vector Machines for multi-class categorisation. The novel couplers suggested in this thesis do not perform significantly different from established couplers. Their advantage is that they are easy to implement and have low run-time requirements. As for the creation of features, the combination of different linguistic units into one feature vector improves the classification rate. Giving more weight to the word sub-vectors is favourable.

Non-probabilistic Support Vector Machines for topic classification are usually thought to work best with linear kernels. Interestingly, experiments presented in this thesis reveal that RBF kernels yield better performance with probabilistic SVMs in some cases. Further research is needed to tell whether this fact is only due to the examined data set. The presented topic classification module is not yet able to update the topic models when topics change or new topics emerge. In the future, adaptation to topic alterations should be considered.

Demonstrator system This thesis does not only investigate the individual modules of the demonstrator, but the performance of the entire system, consisting of speech recogniser, topic segmenter, and topic classifier, as well (Chapter 9 on page 122). The SVM topic classifier outperforms the Naive Bayes classifier. It achieves a micro-averaged F_1 measure of 86 for automatically segmented stories.

Unsupervised Topic Discovery The aforementioned topic classification methods rely on the fact that training data must be provided for every topic to be detected. For the scenario of media monitoring, where the set of customers (those that will receive the media alerts) is known beforehand, a restricted topic list is useful. For media monitoring for e.g. private customers who do not necessarily have a fixed topic profile, the Unsupervised Topic Discovery approach is examined (Chapter 10 on page 127). It does not use any pre-defined topic list at all, but derives the topics from the test data without human interaction. This approach is nearly neglected in research literature, except for the work by Sista. Tests of his system presented in this thesis are based on plain text and automatic transcriptions, as opposed to his plain text only corpus. The UTD approach consists of three steps: preprocessing, initial topic labelling, and final topic labelling (re-classification). In addition to his HMM classifier, a SVM classifier is investigated. The most important conclusion states that for small corpora, the re-classification step delivers bad results and should therefore be omitted. The results of the initial topic labelling are promising, and it is clearly worthwhile to further explore the field of Unsupervised Topic Discovery.

Appendix A

McNemar's statistical test

In order to compare the performance of two classifiers, one can directly compare their evaluation measures (e.g. accuracy, or F_1 measure). But this method does not permit to claim that one classifier does or does not perform significantly better than the other one. Consider, for example, an F_1 measure of 93.4 for classifier A and 93.9 for classifier B. Obviously, classifier B is better, but at the same time the difference in performance is so small that one would hesitate to call it significantly superior.

Statistical tests make it possible to conclude whether two classifiers perform significantly different or not. A null hypothesis has to be defined, and the test tells whether the hypothesis can be rejected or not. This conclusion can only be made with a certain probability of error α, which is usually set a-priory to 0.05 [73].

The null hypothesis to be tested is: The two classifiers will have the same error rate on randomly drawn test samples [22].

Compute the statistic

$$\hat{\chi}^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}$$

using the definitions from Table A.1. If the null hypothesis is correct, the probability that $\hat{\chi}^2$

Table A.1: Contingency table for calculation of the χ^2 measure.

number of test samples misclassified by both A and B = n_{00}	number of test samples misclassified by A but not by B = n_{01}
number of test samples misclassified by B but not by A = n_{10}	number of test samples misclassified by neither A nor B = n_{11}

is greater than $\chi^2_{1,0.95} = 3.841$ is less than 0.05. Thus, if $\hat{\chi}^2$ is greater than $\chi^2_{1,0.95}$, the null hypothesis should be rejected; if it is smaller, the null hypothesis cannot be rejected.

Symbols and abbreviations

Greek symbols

α_i	Lagrange multipliers (model parameters of SVMs)
ξ_i	Slack variables for soft-margin SVMs
λ	Parameters of a Hidden Markov Model
λ	Penalty weight of BIC criterion
μ_{ij}	true pairwise a-posteriori class probability (multi-class pSVM)
Φ	Function for mapping input vectors into high-dimensional space (SVMs)
ϕ	Confidence term for Structural Risk Minimisation

Roman symbols

$\mathbf{A}$	Transition matrix of a HMM, $\mathbf{A} = [a_{ij}]$
a_{ij}	Probability of transition from state s_j to s_i
C	Coefficient used in softmax function
C	Error tolerance for soft-margin SVMs
d	Document
$d(x, y, t)$	Difference of luminance of an image pixel between two consecutive images (audio-visual topic segmentation)
ER	Error rate
f	Number of adjacent frames used for feature creation (topic classification with HMMs)
F_1	Evaluation measure
FA	False alarm rate
fn	Number of false negatives in test set
fp	Number of false positives in test set
h	VC dimension of a set of functions
$h(\mathbf{x})$	Decision function of discriminative classifiers
$I(i)$	Voting counter for class i (multi-class SVMs)

J	Number of prototype vectors used for vector quantisation (VQ)
K	Number of Gaussian mixture components (HMMs)
K	Number of classes
$k(\cdot,\cdot)$	Kernel function
M	Dimension of data points (feature vectors)
M	Miss rate
M_θ	Sequence of prototype labels m
m_j	Index (label) of j-th VQ prototype vector
$m(\mathbf{x})$	Index (label) of the prototype vector $\boldsymbol{\mu}$ that is nearest to $\mathbf{x}$
n	Number of states of a Hidden Markov Model
N	Number of training examples
P	Precision
P_{FA}	Probability of the classifier generating a false alarm
P_{Miss}	Probability of the classifier generating a miss
S	Set of states of a HMM
Q	Set of visited HMM states, $Q = q_1, q_2, \ldots, q_T$
q_t	Active HMM state at time t
R	Recall
r_{ij}	Estimate of μ_{ij} (multi-class pSVM)
T	Number of distinct terms (i.e. vocabulary size) of a document collection
T	Topic
$\hat{T}$	Predicted topic (result of topic classification)
t_i	Term number i
tn	Number of true negatives in test set
tp	Number of true positives in test set
$\tilde{T}$	Sequence of topics T
w	Number of characters in one frame (topic classification with HMMs)
W_i	i-th word in the vocabulary
w_i	i-th word in a document
w_i	Weight of i-th term in feature vector $\mathbf{x}$
w_{word}	Weight of feature sub-vector created from word representation of text
$\mathbf{x}$	Feature vector

$\hat{y}$	Class prediction of a classifier
y_n	Class of data sample n

Abbreviations

ASR	Automatic Speech Recognition
BN	Broadcast News
ch3gram	character 3-gram
DL	Description Length
ERM	Empirical Risk Minimisation
FB	Forward-Backward
HMM	Hidden Markov Model
LH	Likelihood
LM	Language Model
LVCSR	Large Vocabulary Continuous Speech Recognition
MI	Mutual Information
NN	Neural Network
npSVM	non-probabilistic (conventional) Support Vector Machine
pSVM	probabilistic Support Vector Machine
SRM	Structural Risk Minimisation
SVM	Support Vector Machine
TDT	Topic Detection and Tracking (Conference Series)
TREC	Text REtrieval Conference
UTD	Unsupervised Topic Discovery
VQ	Vector quantisation
WER	Word Error Rate

Bibliography

[1] Libsvm. http://www.csie.ntu.edu.tw/˜cjlin/ libsvm/index.html.

[2] Reuters 21758 collection. http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[3] TDT homepage. http://www.nist.gov/speech/tests/tdt/index.htm.

[4] TREC homepage. http://www-nlpir.nist.gov/projects/tv2003/.

[5] Snowball stemmer. http://snowball.tartarus.org, 1996. Perl Script for German Stemming.

[6] The 2001 Topic Detection and Tracking (TDT 2001) Task Definition and Evaluation Plan, 2001.

[7] ALLAN, J., CARBONELL, J., DODDINGTON, G., YAMRON, J., AND YANG, Y. Topic Detection and Tracking Pilot Study Final Report. In *Proc. Broadcast News Transcription and Understranding Workshop* (1998).

[8] BAHLMANN, C., HAASDONK, B., AND BURKHARDT, H. On-line Handwriting Recognition with Support Vector Machines—A Kernel Approach. In *Proc. of the 8th IWFHR* (2002), pp. 49–54.

[9] BAUM, L., AND SELL, G. Growth Transformations for Functions on Manifolds. *Pacific Math J 27* (1968), 211–227.

[10] BELLMAN, R. E. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[11] BESACIER, L., QUENOT, G., AYACHE, S., AND MORARU, D. Video Story Segmentation with Multi-Modal Features: Experiments on TRECVID 2003. In *6th ACM SIGMM International Workshop on Multimedia Information Retrieval* (2004).

[12] BISHOP, C. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[13] BOSER, B. E., GUYON, I., AND VAPNIK, V. A Training Algorithm for Optimal Margin Classifiers. In *Computational Learing Theory* (1992), pp. 144–152.

[14] BROWN, M., GRUNDY, W., LIN, D., CHRISTIANINI, N., SUGNET, C., JR, M., AND HAUSSLER, D. Support Vector Machine Classification of Microarray Gene Expression Data, 1999.

[15] BUCKLEY, C., SALTON, G., ALLAN, J., AND SINGHAL, A. Automatic Query Expansion using SMART: TREC 3. In *An Overview of the Third Text Retrieval Conference (TREC 3)* (1995), D. Harman, Ed., National Institute of Science and Technology, pp. 69–80. Special Publication 500-225.

[16] BURGES, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery 2*, 2 (1998), 121–167.

[17] CHERKASSKY, V., AND MULIER, F. *Learning From Data. Concepts, Theory, and Methods*. Wiley, 1998.

[18] COOLEY, R. Classification of News Stories Using Support Vector Machines. In *IJCAI'99 Workshop on Text Mining* (Stockholm, Sweden, 1999).

[19] DAI, P., IURGEL, U., AND RIGOLL, G. A Novel Feature Combination Approach for Spoken Document Classification with Support Vector Machines. In *Multimedia Information Retrieval Workshop in conjunction with the SIGIR ACM International Conference on Research and Development in Information Retrieval* (2003).

[20] DARSCHIN, W., AND GERHARD, H. Tendenzen im Zuschauerverhalten. Fernsehgewohnheiten und -reichweiten im Jahr 2003. In *Media Perspektiven 4/2004 und 4/2003.* (2003/04).

[21] DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. Maximum Likelihood From Incomplete Data Via the EM Algorithm. In *Journal of the Royal Statistical Society, Series B* (1977), vol. 39, pp. 1–38.

[22] DIETTERICH, T. G. Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms. *Neural Computation 10*, 7 (1998), 1895–1923.

[23] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification*, 2 ed. Wiley, 2000.

[24] DUMAIS, S., PLATT, J., HECKERMAN, D., AND SAHAMI, M. Inductive Learning Algorithms and Representations for Text Categorization. In *Proc. Conference on Information and Knowledge Management (CIKM)* (1998), pp. 148 – 155.

[25] EICKELER, S. *Automatische Bildfolgenanalyse mit statistischen Mustererkennungsverfahren*. PhD thesis, Gerhard-Mercator-Universität Duisburg, 2001.

[26] EICKELER, S., AND MÜLLER, S. Content-Based Video Indexing of TV Broadcast News Using Hidden Markov Models. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Phoenix, Mar. 1999), pp. 2997–3000.

[27] HAMPAPUR, A., JAIN, R., AND WEYMOUTH, T. Digital Video Indexing in Multimedia Systems. In *Workshop on Indexing and Reuse in Multimedia Systems* (Aug. 1994).

[28] HASTIE, T., AND TIBSHIRANI, R. Classification by Pairwise Coupling. *The Annals of Statistics 26* (1998), 451–471.

[29] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. *The Elements of Statistical Learning*. Springer, 2003.

[30] HAYKIN, S. *Neural Networks. A Comprehensive Foundation*. Macmillan, 1999.

[31] HERNÁNDEZ-ABREGO, G., MENENDEZ-PIDAL, X., AND OLORENSHAW, L. Robust and Efficient Confidence Measure for Isolated Command Recognition. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)* (2001).

[32] HSU, C.-W., AND LIN, C.-J. A Comparison of Methods for Multi-Class Support Vector Machines. *IEEE Transactions on Neural Networks 13* (2002), 415–425.

[33] IURGEL, U., MEERMEIER, R., EICKELER, S., AND RIGOLL, G. New Approaches to Audio-Visual Segmentation of TV News for Automatic Topic Retrieval. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Salt Lake City , Utah, May 2001).

[34] IURGEL, U., AND RIGOLL, G. Spoken Document Classification with SVMs Using Linguistic Unit Weighting and Probabilistic Couplers. In *International Conference on Pattern Recognition (ICPR)* (Cambridge, United Kingdom, Aug. 2004).

[35] IURGEL, U., WERNER, S., AND KOSMALA, A. Automatische Auswertung von Radio- und Fernsehnachrichten: Fortschritte in der Spracherkennung und Themenidentifikation. In *13. Konferenz Elektronische Sprachsignalverarbeitung ESSV* (Dresden, Germany, Sept. 2002).

[36] IURGEL, U., WERNER, S., KOSMALA, A., WALLHOFF, F., AND RIGOLL, G. Audio-Visual Analysis of Multimedia Documents for Automatic Topic Identification. In *Signal Processing, Pattern Recognition, and Applications* (Crete, Greece, June 2002), pp. 550–555. ACTA Press.

[37] IURGEL, U., WERNER, S., AND RIGOLL, G. Vergleich von automatischer und manueller Segmentierung von Fernsehnachrichten und deren Einfluss auf die Sprach- und Themenerkennung. In *14. Konferenz Elektronische Sprachsignalverarbeitung ESSV* (Karlsruhe, Germany, Sept. 2003).

[38] JELINEK, F. Continuous Speech Recognition by Statistical Methods. *Proceedings of the IEEE 64*, 4 (Apr. 1976), 532–556.

[39] JELINEK, F. *Statistical methods for speech recognition*. MIT Press, 1997.

[40] JIN, H., SCHWARTZ, R., SISTA, S., AND WALLS, F. Topic Tracking for Radio, TV Broadcast, and Newswire. In *Proceedings of the DARPA Broadcast News Workshop* (1999).

[41] JOACHIMS, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *University of Dortmund, LS-8 Report 23* (1997).

[42] JOACHIMS, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (1998), Springer Verlag, Heidelberg, pp. 137–142.

[43] JOACHIMS, T. *Learning to Classify Text Using Support Vector Machines*. Kluwer, 2002.

[44] JOACHIMS, T. Tutorial on Support Vector Machines. International Conference on Research and Development in Information Retrieval (SIGIR), July 2003.

[45] LARSON, M., EICKELER, S., PAASS, G., LEOPOLD, E., AND KINDERMANN, J. Exploring Sub-Word Features and Linear Support Vector Machines for German Spoken Document Classification. In *ICSLP* (2002).

[46] LEOPOLD, E., AND KINDERMANN, J. Text Categorization with Support Vector Machines. How to Represent Texts in Input Space? In *Machine Learning* (2002), vol. 46, pp. 423–444.

[47] LEOPOLD, E., MAY, M., AND PAASS, G. Data Mining and Text Mining for Science & Technology Research. In *Handbook of Quantitative Science and Technology Research*, H. F. Moed, W. Glänzel, and U. Schmoch, Eds. Kluwer, 2004.

[48] LEWIS, D. D. Evaluating and Optimizing Autonomous Text Classification Systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, 1995), E. A. Fox, P. Ingwersen, and R. Fidel, Eds., ACM Press, pp. 246–254.

[49] LIN, H.-T., LIN, C.-J., AND WENG, R. A Note on Platt's Probabilistic Outputs for Support Vector Machines. In *Neural Information Processing Systems 2003* (2003).

[50] LO, Y.-Y., AND GAUVAIN, J.-L. The LIMSI Topic Tracking System for TDT2001. In *Proc. TDT'01* (Gaithersburg, Nov. 2001).

[51] LOONEY, C. G. *Pattern Recognition Using Neural Networks*. Oxford University Press, 1997.

[52] MANNING, C. D., AND SCHÜTZE, H. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge and London, 1999.

[53] MAYFIELD, J. Introduction to Information Retrieval. CLSP Workshop Seminar, 2002. http://www.clsp.jhu.edu/ws2002/preworkshop/mayfield.pdf.

[54] MILLER, D. R. H., LEEK, T., AND SCHWARTZ, R. M. A Hidden Markov Model Information Retrieval System. In *Proceedings of SIGIR1999, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), pp. 214–221.

[55] NEUKIRCHEN, C. *Integration neuronaler Vektorquantisierer in ein Hidden-Markov-Modell-basiertes System zur automatischen Spracherkennung*. PhD thesis, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Germany, 1999.

[56] NEUKIRCHEN, C., AND RIGOLL, G. Training of MMI Neural Networks As Vector Quantizers. Tech. rep., Department of Computer Science, University of Duisburg, Germany, Mar. 1996. http://www.fb9-ti.uni-duisburg.de/publ/96/internal.01.ps.gz.

[57] NEUKIRCHEN, C., ROTTLAND, J., WILLETT, D., AND RIGOLL, G. A Continuous Density Interpretation of Discrete HMM Systems and MMI-Neural Networks. *IEEE Transactions on Speech and Audio Processing 9*, 4 (2001), 367–377.

[58] PAASS, G., LEOPOLD, E., LARSON, M., KINDERMANN, J., AND EICKELER, S. SVM Classification Using Sequences of Phonemes and Syllables. In *European Conference on Machine Learning (ECML)* (2002).

[59] PAPAGEORGIOU, C., OREN, M., AND POGGIO, T. A General Framework for Object Detection. In *Computer Vision, 1998. Sixth International Conference on* (1998), pp. 555–562.

[60] PAPKA, R. *On-line New Event Detection, Clustering, and Tracking*. PhD thesis, University of Massachusetts, Amherst, 1999.

[61] PFEIFER, U. Text::german - German Grundform Reduction (Perl script).

http://search.cpan.org/˜ulpfr/Text-German-0.03/, 1996. Perl Script for German Stemming.

[62] PLATT, J. Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods. In *Advances in Large Margin Classiers,*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. MIT Press, 1999.

[63] PRICE, D., KNER, S., PERSONNAZ, L., AND DREYFUS, G. Pairwise Neural Network Classifiers with Probabilistic Outputs. In *Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds. MIT Press, 1995.

[64] RABINER, L. R. A Tutorial on HMM and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77*, 2 (Feb. 1989), 257–286.

[65] RABINER, L. R. *Fundamentals of Speech Recognition*. Prentice Hal, 1993.

[66] RABINER, L. R., AND JUANG, B. H. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine* (January 1986), 4–16.

[67] RENNIE, J., SHIH, L., TEEVAN, J., AND KARGER, D. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning* (2003).

[68] RENNIE, J. D. M., AND RIFKIN, R. Improving Multiclass Text Classification with the Support Vector Machine. Tech. rep., Massachusetts Institute of Technolgy, AI Memo AIM-2001-026, Oct. 2001.

[69] RIEDMILLER, M., AND BRAUN, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks (ICNN)* (1993), pp. 586–591.

[70] RIGOLL, G. Maximum Mutual Information Neural Networks for Hybrid Connectionist-HMM Speech Recognition Systems. *IEEE Transactions on Speech and Audio Processing, Special Issue on Neural Networks for Speech 2*, 1 (1994), 175–184.

[71] RIGOLL, G., KOSMALA, A., ROTTLAND, J., AND NEUKIRCHEN, C. A Comparison Between Continuous and Discrete Density Hidden Markov Models for Cursive Handwriting Recognition. In *Int. Conference on Pattern Recognition (ICPR)* (Vienna, Aug. 1996), vol. 2, pp. 205–209.

[72] ROSENFELD, R. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

[73] SACHS, L. *Angewandte Statistik*, 9 ed. Springer, 1999.

[74] SALTON, G., AND BUCKLEY, C. Term-Weighing Approaches in Automatic Text Retrieval. In *Information Processing & Management* (1988), vol. 24, pp. 513–523.

[75] SCHÖLKOPF, B. *Support Vector Learning*. PhD thesis, TU Berlin, 1997.

[76] SCHÖLKOPF, B., BURGES, C., AND SMOLA, A. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998.

[77] SCHÖLKOPF, B., AND SMOLA, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[78] SCHUKAT-TALAMAZZINI, E. G. *Automatische Spracherkennung – Grundlagen, statistische Modelle und effiziente Algorithmen*. Künstliche Intelligenz. Vieweg, Braunschweig, 1995.

[79] SCHWARTZ, R., IMAI, T., KUBALA, F., NGUYEN, L., AND MAKHOUL, J. A Maximum Likelihood Model for Topic Classification of Broadcast News. In *Proceedings of the European Conference On Speech Communication and Technology* (Rhodes, Greece, Sept. 1997), pp. 1455–1458.

[80] SCHWARTZ, R., SISTA, S., AND LEEK, T. Unsupervised Topic Discovery. In *Proc. Workshop on Language Modeling and Information Retrieval* (2001), pp. 72–77.

[81] SCHWARZ, G. Estimating the Dimension of a Model. *The Annals of Statistics 6*, 2 (1978), 461–464.

[82] SISTA, S. *Statistical Methods for Unsupervised Topic Discovery*. PhD thesis, Northeastern University Boston, Massachusetts, USA, 2002.

[83] SISTA, S., SCHWARTZ, R., LEEK, T. R., AND MAKHOUL, J. An Algorithm for Unsupervised Topic Discovery from Broadcast News Stories. In *Proc. Human Language Technology Conference (HLT), San Diego, USA* (2002).

[84] SMEATON, A. F., KRAAIJ, W., AND OVER, P. TRECVID 2003 - An Overview, 2003. http://www-nlpir.nist.gov/projects/tvpubs/tvpapers03/tv3overview.pdf.

[85] TAKAO, S., OGATA, J., AND ARIKI, Y. Topic Segmentation of News Speech Using Word Similarity. In *ACM Multimedia* (2000), pp. 442–444.

[86] THEODORIDIS, S., AND KOUTROUMBAS, K. *Pattern Recognition*. Academic Press, 1999.

[87] TRITSCHLER, A., AND GOPINATH, R. Improved Speaker Segmentation and Segments Clustering Using the Bayesian Information Criterion. In *Proc. EUROSPEECH* (1999), vol. 2, pp. 679–682.

[88] VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Springer, 1995.

[89] VITERBI, A. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory 13*, 2 (1967), 260–269.

[90] WAHLSTER, W. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, 2000.

[91] WEBB, A. *Statistical Pattern Recognition*, 2 ed. Wiley, 2002.

[92] WERNER, S., IURGEL, U., KOSMALA, A., AND RIGOLL, G. Automatic Topic Identification in Multimedia Broadcast Data. In *IEEE International Conference on Multimedia and Expo (ICME)* (Lausanne, Switzerland, Aug. 2002).

[93] WILLETT, D. *Beiträge zur statistischen Modellierung und effizienten Dekodierung in der automatischen Spracherkennung*. PhD thesis, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Germany, Nov. 2000.

[94] WILLETT, D., NEUKIRCHEN, C., AND RIGOLL, G. Ducoder - the Duisburg University LVCSR Stackdecoder. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (Istanbul, Turkey, June 2000), pp. 1555–1558.

[95] WU, T.-F., LIN, C.-J., AND WENG, R. C. Probability Estimates for Multi-Class Classification by Pairwise Coupling. www.csie.ntu.edu.tw/˜cjlin/papers/svmprob/svmprob.pdf, September 2003.

[96] YAMATO, J., OHYA, J., AND ISHII, K. Recognizing Human Action in Time-Sequential Images Using Hidden Markov Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Champaign, USA, June 1992), pp. 379–385.

[97] YANG, Y. An Evaluation of Statistical Approaches to Text Categorization. Tech. rep., CMU-CS-97-12, Carnegie Mellon University, 1997.

[98] YANG, Y. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval 1*, 1-2 (1999), 67–88.

[99] YANG, Y. A Study on Thresholding Strategies for Text Categorization. In *Proc. 24th Intl. Conference on Research and Development in Information Retrieval (SIGIR) 01* (New Orleans, US, 2001), pp. 137–145.

[100] YANG, Y., CARBONELL, J., BROWN, R., PIERCE, T., ARCHIBALD, B., AND LIU, X. Learning Approaches for Detecting and Tracking News Events. *IEEE Intelligent Systems: Special Issue on Applications of Intelligent Information Retrieval 14*, 4 (1999), 32–43.

[101] YANG, Y., AND LIU, X. A Re-Examination of Text Categorization Methods. In *22nd Annual International SIGIR* (Berkley, August 1999), pp. 42–49.

[102] ZOBEL, J., AND DART, P. W. Phonetic String Matching: Lessons from Information Retrieval. In *Proc. 19th Intl. Conference on Research and Development in Information Retrieval (SIGIR) 96* (1996), ACM Press, pp. 166–172.