A Brief Survey of Text Mining

Andreas Hotho
KDE Group
University of Kassel
hotho@cs.uni-kassel.de

Andreas Nürnberger
Information Retrieval Group
School of Computer Science
Otto-von-Guericke-University Magdeburg
nuernb@iws.cs.uni-magdeburg.de

Gerhard Paaß
Fraunhofer AiS
Knowledge Discovery Group
Sankt Augustin
gerhard.paass@ais.fraunhofer.de

May 13, 2005

Abstract

The enormous amount of information stored in unstructured texts cannot simply be used for further processing by computers, which typically handle text as simple sequences of character strings. Therefore, specific (pre-)processing methods and algorithms are required in order to extract useful patterns. Text mining refers generally to the process of extracting interesting information and knowledge from unstructured text. In this article, we discuss text mining as a young and interdisciplinary field in the intersection of the related areas information retrieval, machine learning, statistics, computational linguistics and especially data mining. We describe the main analysis tasks preprocessing, classification, clustering, information extraction and visualization. In addition, we briefly discuss a number of successful applications of text mining.

1 Introduction

As computer networks become the backbones of science and economy enormous quantities of machine readable documents become available. There are estimates that 85% of business information lives in the form of text [TMS05]. Unfortunately, the usual logic-based programming paradigm has great difficulties in capturing the fuzzy and often ambiguous relations in text documents. Text mining aims at disclosing the concealed information by means of methods which on the one hand are able to cope with the large number of words and structures in natural language and on the other hand allow to handle vagueness, uncertainty and fuzziness.

In this paper we describe text mining as a truly interdisciplinary method drawing on information retrieval, machine learning, statistics, computational linguistics and especially data mining. We first give a short sketch of these methods and then define text mining in relation to them. Later sections survey state of the art approaches for the main analysis tasks preprocessing, classification, clustering, information extraction and visualization. The last section exemplifies text mining in the context of a number of successful applications.

1.1 Knowledge Discovery

In literature we can find different definitions of the terms knowledge discovery or knowledge discovery in databases (KDD) and data mining. In order to distinguish data mining from KDD we define KDD according to Fayyad as follows [FPSS96]:

> ”Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”

The analysis of data in KDD aims at finding hidden patterns and connections in these data. By data we understand a quantity of facts, which can be, for instance, data in a database, but also data in a simple text file. Characteristics that can be used to measure the quality of the patterns found in the data are the comprehensibility for humans, validity in the context of given statistic measures, novelty and usefulness. Furthermore, different methods are able to discover not only new patterns but to produce at the same time generalized models which represent the found connections. In this context, the expression “potentially useful” means that the samples to be found for an application generate a benefit for the user. Thus the definition couples knowledge discovery with a specific application.

Knowledge discovery in databases is a process that is defined by several processing steps that have to be applied to a data set of interest in order to extract useful patterns. These steps have to be performed iteratively and several steps usually require interactive feedback from a user. As defined by the CRoss Industry Standard Process for Data Mining (Crisp DM[1]) model [cri99] the main steps are: (1) business understanding[2], (2) data understanding, (3) data preparation, (4) modelling, (5) evaluation, (6) deployment (cf. fig. 1[3]). Besides the initial problem of analyzing and understanding the overall task (first two steps) one of the most time consuming steps is data preparation. This is especially of interest for text mining which needs special preprocessing methods to

[1] `http://www.crisp-dm.org/`

[2] Business understanding could be defined as understanding the problem we need to solve. In the context of text mining, for example, that we are looking for groups of similar documents in a given document collection.

[3] figure is taken from `http://www.crisp-dm.org/Process/index.htm`

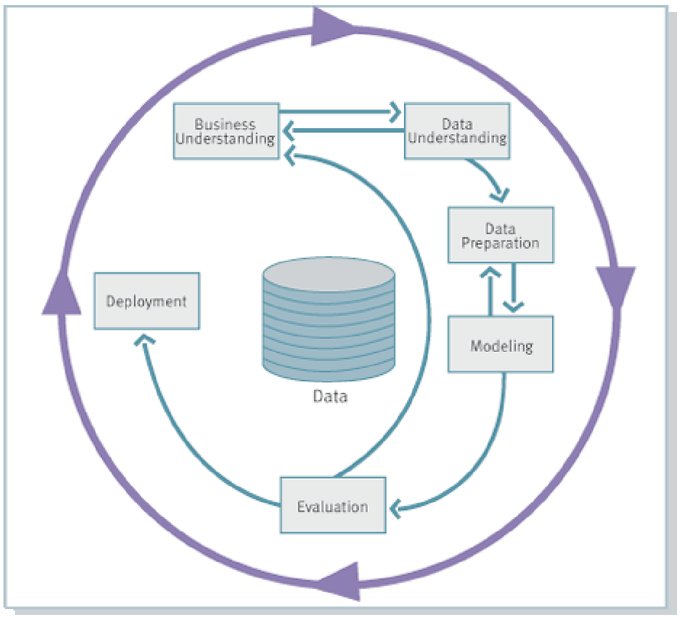


Figure 1: Phases of Crisp DM

convert textual data into a format which is suitable for data mining algorithms. The application of data mining algorithms in the modelling step, the evaluation of the obtained model and the deployment of the application (if necessary) are closing the process cycle. Here the modelling step is of main interest as text mining frequently requires the development of new or the adaptation of existing algorithms.

1.2 Data Mining, Machine Learning and Statistical Learning

Research in the area of data mining and knowledge discovery is still in a state of great flux. One indicator for this is the sometimes confusing use of terms. On the one side there is *data mining as synonym for KDD*, meaning that data mining contains all aspects of the knowledge discovery process. This definition is in particular common in practice and frequently leads to problems to distinguish the terms clearly. The second way of looking at it considers *data mining as part of the KDD-Processes* (see [FPSS96]) and describes the modelling phase, i.e. the application of algorithms and methods for the calculation of the searched patterns or models. Other authors like for instance Kumar and Joshi [KJ03] consider data mining in addition as the search for valuable information in *large quantities of data*. In this article, we equate data mining with the modelling phase of the KDD process.

The roots of data mining lie in most diverse areas of research, which underlines the interdisciplinary character of this field. In the following we briefly discuss the relations to three of the addressed research areas: Databases, machine learning and statistics.

Databases are necessary in order to analyze large quantities of data efficiently. In this connection, a database represents not only the medium for consistent storing and accessing, but moves in the closer interest of research, since the analysis of the data with data mining algorithms can be supported by databases and thus the use of database technology in the data mining process might be useful. An overview of data mining from the database perspective can be found in [CHY96].

Machine Learning (ML) is an area of artificial intelligence concerned with the development of techniques which allow computers to ”learn” by the analysis of data sets. The focus of most machine learning methods is on symbolic data. ML is also concerned with the algorithmic complexity of computational implementations. Mitchell presents many of the commonly used ML methods in [Mit97].

Statistics has its grounds in mathematics and deals with the science and practice for the analysis of empirical data. It is based on statistical theory which is a branch of applied mathematics. Within statistical theory, randomness and uncertainty are modelled by probability theory. Today many methods of statistics are used in the field of KDD. Good overviews are given in [HTF01, Be99, Mai02].

1.3 Definition of Text Mining

Text mining or knowledge discovery from text (KDT) — for the first time mentioned in Feldman et al. [FD95] — deals with the machine supported analysis of text. It uses techniques from information retrieval, information extraction as well as natural language processing (NLP) and connects them with the algorithms and methods of KDD, data mining, machine learning and statistics. Thus, one selects a similar procedure as with the KDD process, whereby not data in general, but text documents are in focus of the analysis. From this, new questions for the used data mining methods arise. One problem is that we now have to deal with problems of — from the data modelling perspective — unstructured data sets.

If we try to define text mining, we can refer to related research areas. For each of them, we can give a different definition of text mining, which is motivated by the specific perspective of the area:

Text Mining = Information Extraction. The first approach assumes that text mining essentially corresponds to information extraction (cf. section 3.3) — the extraction of facts from texts.

Text Mining = Text Data Mining. Text mining can be also defined — similar to data mining — as the application of algorithms and methods from the fields machine learning and statistics to texts with the goal of finding useful patterns. For this purpose it is necessary to pre-process the texts accordingly. Many authors use information extraction methods, natural language processing or some simple preprocessing steps in order to extract data from texts. To the extracted data then data mining algorithms can be applied (see [NM02, Gai03]).

Text Mining = KDD Process. Following the knowledge discovery process model [cri99], we frequently find in literature text mining as a process with a series of partial steps, among other things also information extraction as well as the use of data mining or statistical procedures. Hearst summarizes this in [Hea99] in a general manner as the extraction of not yet discovered information in large collections of texts. Also Kodratoff in [Kod99] and Gomez in [Hid02] consider text mining as process orientated approach on texts.

In this article, we consider text mining mainly as text data mining. Thus, our focus is on methods that extract useful patterns from texts in order to, e.g., categorize or structure text collections or to extract useful information.

1.4 Related Research Areas

Current research in the area of text mining tackles problems of text representation, classification, clustering, information extraction or the search for and modelling of hidden patterns. In this context the selection of characteristics and also the influence of domain knowledge and domain-specific procedures plays an important role. Therefore, an adaptation of the known data mining algorithms to text data is usually necessary. In order to achieve this, one frequently relies on the experience and results of research in information retrieval, natural language processing and information extraction. In all of these areas we also apply data mining methods and statistics to handle their specific tasks:

Information Retrieval (IR). Information retrieval is the finding of documents which contain answers to questions and not the finding of answers itself [Hea99]. In order to achieve this goal statistical measures and methods are used for the automatic processing of text data and comparison to the given question. Information retrieval in the broader sense deals with the entire range of information processing, from data retrieval to knowledge retrieval (see [SJW97] for an overview). Although, information retrieval is a relatively old research area where first attempts for automatic indexing where made in 1975 [SWY75], it gained increased attention with the rise of the World Wide Web and the need for sophisticated search engines.

Even though, the definition of information retrieval is based on the idea of questions and answers, systems that retrieve documents based on keywords, i.e. systems that perform *document retrieval* like most search engines, are frequently also called information retrieval systems.

Natural Language Processing (NLP). The general goal of NLP is to achieve a better understanding of natural language by use of computers [Kod99]. Others include also the employment of simple and durable techniques for the fast processing of text, as they are presented e.g. in [Abn91]. The range of the assigned techniques reaches from the simple manipulation of strings to the automatic processing of natural language inquiries. In addition, linguistic analysis techniques are used among other things for the processing of text.

Information Extraction (IE). The goal of information extraction methods is the extraction of specific information from text documents. These are stored in data base-like patterns (see [Wil97]) and are then available for further use. For further details see section 3.3.

In the following, we will frequently refer to the above mentioned related areas of research. We will especially provide examples for the use of machine learning methods in information extraction and information retrieval.

2 Text Encoding

For mining large document collections it is necessary to pre-process the text documents and store the information in a data structure, which is more appropriate for further processing than a plain text file. Even though, meanwhile several methods exist that try to exploit also the syntactic structure and semantics of text, most text mining approaches are based on the idea that a text document can be represented by a set of words, i.e. a text document is described based on the set of words contained in it (*bag-of-words* representation). However, in order to be able to define at least the importance of a word within a given document, usually a vector representation is used, where for each word a numerical ”importance” value is stored. The currently predominant approaches based on this idea are the vector space model [SWY75], the probabilistic model [Rob77] and the logical model [van86].

In the following we briefly describe, how a bag-of-words representation can be obtained. Furthermore, we describe the vector space model and corresponding similarity measures in more detail, since this model will be used by several text mining approaches discussed in this article.

2.1 Text Preprocessing

In order to obtain all words that are used in a given text, a *tokenization* process is required, i.e. a text document is split into a stream of words by removing all punctuation marks and by replacing tabs and other non-text characters by single white spaces. This tokenized representation is then used for further processing. The set of different words obtained by merging all text documents of a collection is called the *dictionary* of a document collection.

In order to allow a more formal description of the algorithms, we define first some terms and variables that will be frequently used in the following: Let D be the set of documents and $T = \{t_1, \dots, t_m\}$ be the dictionary, i.e. the set of all different terms occurring in D, then the absolute frequency of term $t \in T$ in document $d \in D$ is given by $\mathrm{tf}(d, t)$. We denote the term vectors $\vec{t_d} = (\mathrm{tf}(d, t_1), \dots, \mathrm{tf}(d, t_m))$. Later on, we will also need the notion of the centroid of a set X of term vectors. It is defined as the mean value $\vec{t_X} := \frac{1}{|X|} \sum_{\vec{t_d} \in X} \vec{t_d}$ of its term vectors. In the sequel, we will apply tf also on subsets of terms: For $T' \subseteq T$, we let $\mathrm{tf}(d, T') := \sum_{t \in T'} \mathrm{tf}(d, t)$.

2.1.1 Filtering, Lemmatization and Stemming

In order to reduce the size of the dictionary and thus the dimensionality of the description of documents within the collection, the set of words describing the documents can be reduced by filtering and lemmatization or stemming methods.

Filtering methods remove words from the dictionary and thus from the documents. A standard filtering method is stop word filtering. The idea of stop word filtering is to remove words that bear little or no content information, like articles, conjunctions, prepositions, etc. Furthermore, words that occur extremely often can be said to be of little information content to distinguish between documents, and also words that occur very seldom are likely to be of no particular statistical relevance and can be removed from the dictionary [FBY92]. In order to further reduce the number of words in the dictionary, also (index) term selection methods can be used (see Sect. 2.1.2).

Lemmatization methods try to map verb forms to the infinite tense and nouns to the singular form. However, in order to achieve this, the word form has to be known, i.e. the part of speech of every word in the text document has to be assigned. Since this tagging process is usually quite time consuming and still error-prone, in practice frequently stemming methods are applied.

Stemming methods try to build the basic forms of words, i.e. strip the plural 's' from nouns, the 'ing' from verbs, or other affixes. A stem is a natural group of words with equal (or very similar) meaning. After the stemming process, every word is represented by its stem. A well-known rule based stemming algorithm has been originally proposed by Porter [Por80]. He defined a set of production rules to iteratively transform (English) words into their stems.

2.1.2 Index Term Selection

To further decrease the number of words that should be used also indexing or keyword selection algorithms can be used (see, e.g. [DDFL90, WMB99]). In this case, only the selected keywords are used to describe the documents. A simple method for keyword selection is to extract keywords based on their entropy. E.g. for each word t in the vocabulary the entropy as defined by [LS89] can be computed:

$$W(t) = 1 + \frac{1}{\log_2 |D|} \sum_{d \in D} P(d,t) \log_2 P(d,t) \quad \text{with} \quad P(d,t) = \frac{\mathrm{tf}(d,t)}{\sum_{l=1}^{n} \mathrm{tf}(d_l,t)} \tag{1}$$

Here the entropy gives a measure how well a word is suited to separate documents by keyword search. For instance, words that occur in many documents will have low entropy. The entropy can be seen as a measure of the importance of a word in the given domain context. As index words a number of words that have a high entropy relative to their overall frequency can be chosen, i.e. of words occurring equally often those with the higher entropy can be preferred.

In order to obtain a fixed number of index terms that appropriately cover the documents, a simple greedy strategy can be applied: From the first document in the collection select the term with the highest relative entropy (or information gain as described in Sect. 3.1.1) as an index term. Then mark this document and all other documents containing this term. From the first of the remaining unmarked documents select again the term with the highest relative entropy as an index term. Then mark again this document and all other documents containing this term. Repeat this process until all documents are marked, then unmark them all and start again. The process can be terminated when the desired number of index terms have been selected. A more detailed discussion of the benefits of this approach for clustering - with respect to reduction of words required in order to obtain a good clustering performance - can be found in [BN04].

An index term selection methods that is more appropriate if we have to learn a classifier for documents is discussed in Sect. 3.1.1. This approach also considers the word distributions within the classes.

2.2 The Vector Space Model

Despite of its simple data structure without using any explicit semantic information, the vector space model enables very efficient analysis of huge document collections. It was originally introduced for indexing and information retrieval [SWY75] but is now used also in several text mining approaches as well as in most of the currently available document retrieval systems.

The vector space model represents documents as vectors in m-dimensional space, i.e. each document *d* is described by a numerical feature vector $w(d) = (x(d, t_1), \ldots, x(d, t_m))$. Thus, documents can be compared by use of simple vector operations and even queries can be performed by encoding the query terms similar to the documents in a query vector. The query vector can then be compared to each document and a result list can be obtained by ordering the documents according to the computed similarity [SAB94]. The main task of the vector space representation of documents is to find an appropriate encoding of the feature vector.

Each element of the vector usually represents a word (or a group of words) of the document collection, i.e. the size of the vector is defined by the number of words (or groups of words) of the complete document collection. The simplest way of document encoding is to use binary term vectors, i.e. a vector element is set to one if the corresponding word is used in the document and to zero if the word is not. This encoding will result in a simple Boolean comparison or search if a query is encoded in a vector. Using Boolean encoding the importance of all terms for a specific query or comparison is considered as similar. To improve the performance usually term weighting schemes are used, where the weights reflect the importance of a word in a specific document of the considered collection. Large weights are assigned to terms that are used frequently in relevant documents but rarely in the whole document collection [SB88]. Thus a weight $w(d, t)$ for a term t in document *d* is computed by term frequency tf(d, t) times inverse document frequency idf(t), which describes the term specificity within the document collection. In [SAB94] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency — defined as $idf(t) := log(N/n_t)$ —, a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$w(d, t) = \frac{\mathrm{tf}(d, t) \log(N/n_t)}{\sqrt{\sum_{j=1}^{m} tf(d, t_j)^2 (\log(N/n_{t_j}))^2}}, \tag{2}$$

where N is the size of the document collection D and n_t is the number of documents in D that contain term t.

Based on a weighting scheme a document *d* is defined by a vector of term weights $w(d) = (w(d,t_1),\dots,w(d,t_m))$ and the similarity S of two documents d_1 and d_2 (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors (by which – if we assume normalized vectors – the cosine between the two document vectors is computed), i.e.

$$S(d_1,d_2) = \sum\nolimits_{k=1}^{m} w(d_1,t_k)\cdot w(d_2,t_k). \tag{3}$$

A frequently used distance measure is the Euclidian distance. We calculate the distance between two text documents $d_1, d_2 \in D$ as follows:

$$dist(d_1,d_2) = \sqrt[2]{\sum\nolimits_{k=1}^{m} |w(d_1,t_k) - w(d_2,t_k)|^2}\ . \tag{4}$$

However, the Euclidean distance should only be used for normalized vectors, since otherwise the different lengths of documents can result in a smaller distance between documents that share less words than between documents that have more words in common and should be considered therefore as more similar.

Note that for normalized vectors the scalar product is not much different in behavior from the Euclidean distance, since for two vectors $\vec{x}$ and $\vec{y}$ it is

$$\cos\varphi = \frac{\vec{x}\vec{y}}{|\vec{x}|\cdot|\vec{y}|} = 1 - \frac{1}{2}\,d^2\left(\frac{\vec{x}}{|\vec{x}|},\frac{\vec{y}}{|\vec{y}|}\right).$$

For a more detailed discussion of the vector space model and weighting schemes see, e.g. [BYRN99, Gre98, SB88, SWY75].

2.3 Linguistic Preprocessing

Often text mining methods may be applied without further preprocessing. Sometimes, however, additional linguistic preprocessing (c.f. [MS01a]) may be used to enhance the available information about terms. For this, the following approaches are frequently applied:

Part-of-speech tagging (POS) determines the part of speech tag, e.g. noun, verb, adjective, etc. for each term.

Text chunking aims at grouping adjacent words in a sentence. An example of a chunk is the noun phrase “the current account deficit”.

Word Sense Disambiguation (WSD) tries to resolve the ambiguity in the meaning of single words or phrases. An example is ‘bank’ which may have – among others – the senses ‘financial institution’ or the ‘border of a river or lake’. Thus, instead of terms the specific meanings could be stored in the vector space representation. This leads to a bigger dictionary but considers the semantic of a term in the representation.

Parsing produces a full parse tree of a sentence. From the parse, we can find the relation of each word in the sentence to all the others, and typically also its function in the sentence (e.g. subject, object, etc.).

Linguistic processing either uses lexica and other resources as well as hand-crafted rules. If a set of examples is available machine learning methods as described in section 3, especially in section 3.3, may be employed to learn the desired tags.

It turned out, however, that for many text mining tasks linguistic preprocessing is of limited value compared to the simple bag-of-words approach with basic preprocessing. The reason is that the co-occurrence of terms in the vector representation serves as an automatic disambiguation, e.g. for classification [LK02]. Recently some progress was made by enhancing bag of words with linguistic feature for text clustering and classification [HSS03, BH04].

3 Data Mining Methods for Text

One main reason for applying data mining methods to text document collections is to structure them. A structure can significantly simplify the access to a document collection for a user. Well known access structures are library catalogues or book indexes. However, the problem of manual designed indexes is the time required to maintain them. Therefore, they are very often not up-to-date and thus not usable for recent publications or frequently changing information sources like the World Wide Web. The existing methods for structuring collections either try to assign keywords to documents based on a given keyword set (classification or categorization methods) or automatically structure document collections to find groups of similar documents (clustering methods). In the following we first describe both of these approaches. Furthermore, we discuss in Sect. 3.3 methods to automatically extract useful information patterns from text document collections. In Sect. 3.4 we review methods for visual text mining. These methods allow in combination with structuring methods the development of powerful tools for the interactive exploration of document collections. We conclude this section with a brief discussion of further application areas for text mining.

3.1 Classification

Text classification aims at assigning pre-defined classes to text documents [Mit97]. An example would be to automatically label each incoming news story with a topic like ”sports”, ”politics”, or ”art”. Whatever the specific method employed, a data mining classification task starts with a *training set* $D = (d_1, \ldots, d_n)$ of documents that are already labelled with a class $L \in \mathbb{L}$ (e.g. sport, politics). The task is then to determine a *classification model*

$$f : D \rightarrow \mathbb{L} \qquad f(d) = L \tag{5}$$

which is able to assign the correct class to a new document *d* of the domain.

To measure the performance of a classification model a random fraction of the labelled documents is set aside and not used for training. We may classify the documents of this *test set* with the classification model and compare the estimated labels with the true labels. The fraction of correctly classified documents in relation to the total number of documents is called *accuracy* and is a first performance measure.

Often, however, the target class covers only a small percentage of the documents. Then we get a high accuracy if we assign each document to the alternative class. To avoid this effect different measures of classification success are often used. *Precision* quantifies the fraction of retrieved documents that are in fact relevant, i.e. belong to the target class. *Recall* indicates which fraction of the relevant documents is retrieved.

$$\text{precision} = \frac{\#\{\text{relevant} \cap \text{retrieved}\}}{\#\text{retrieved}} \qquad \text{recall} = \frac{\#\{\text{relevant} \cap \text{retrieved}\}}{\#\text{relevant}} \tag{6}$$

Obviously there is a trade off between precision and recall. Most classifiers internally determine some "degree of membership" in the target class. If only documents of high degree are assigned to the target class, the precision is high. However, many relevant documents might have been overlooked, which corresponds to a low recall. When on the other hand the search is more exhaustive, recall increases and precision goes down. The *F-score* is a compromise of both for measuring the overall performance of classifiers.

$$F = \frac{2}{1/\text{recall} + 1/\text{precision}} \tag{7}$$

3.1.1 Index Term Selection

As document collections often contain more than 100000 different words we may select the most informative ones for a specific classification task to reduce the number of words and thus the complexity of the classification problem at hand. One commonly used ranking score is the *information gain* which for a term t_j is defined as

$$IG(t_j) = \sum_{c=1}^{2} p(L_c) \log_2 \frac{1}{p(L_c)} - \sum_{m=0}^{1} p(t_j{=}m) \sum_{c=1}^{2} p(L_c|t_j{=}m) \log_2 \frac{1}{p(L_c|t_j{=}m)} \tag{8}$$

Here $p(L_c)$ is the fraction of training documents with classes L_1 and L_2, $p(t_j{=}1)$ and $p(t_j{=}0)$ is the number of documents with / without term t_j and $p(L_c|t_j{=}m)$ is the conditional probability of classes L_1 and L_2 if term t_j is contained in the document or is missing. It measures how useful t_j is for predicting L_1 from an information-theoretic point of view. We may determine $IG(t_j)$ for all terms and remove those with very low information gain from the dictionary.

In the following sections we describe the most frequently used data mining methods for text categorization.

3.1.2 Naïve Bayes Classifier

Probabilistic classifiers start with the assumption that the words of a document d_i have been generated by a probabilistic mechanism. It is supposed that the class $L(d_i)$ of document d_i has some relation to the words which appear in the document. This may be described by the conditional distribution $p(t_1, \ldots, t_{n_i}|L(d_i))$ of the n_i words given the class. Then the *Bayesian formula* yields the probability of a class given the words of a document [Mit97]

$$p(L_c|t_1, \ldots, t_{n_i}) = \frac{p(t_1, \ldots, t_{n_i}|L_c)p(L_c)}{\sum_{L \in \mathbb{L}} p(t_1, \ldots, t_{n_i}|L)p(L)}$$

Note that each document is assumed to belong to exactly one of the k classes in $\mathbb{L}$. The prior probability $p(L)$ denotes the probability that an arbitrary document belongs to class L before its words are known. Often the prior probabilities of all classes may be taken to be equal. The conditional probability on the left is the desired *posterior probability* that the document with words $t_1, \ldots, t_{n_i}$ belongs to class L_c. We may assign the class with highest posterior probability to our document.

For document classification it turned out that the specific order of the words in a document is not very important. Even more we may assume that for documents of a given class a word appears in the document irrespective of the presence of other words. This leads to a simple formula for the conditional probability of words given a class L_c

$$p(t_1, \ldots, t_{n_i}|L_c) = \prod_{j=1}^{n_i} p(t_j|L_c)$$

Combining this “naïve” independence assumption with the Bayes formula defines the *Naïve Bayes classifier* [Goo65]. Simplifications of this sort are required as many thousand different words occur in a corpus.

The naïve Bayes classifier involves a learning step which simply requires the estimation of the probabilities of words $p(t_j|L_c)$ in each class by its relative frequencies in the documents of a training set which are labelled with L_c. In the classification step the estimated probabilities are used to classify a new instance according to the Bayes rule. In order to reduce the number of probabilities $p(t_j|L_m)$ to be estimated, we can use index term selection methods as discussed above in Sect. 3.1.1.

Although this model is unrealistic due to its restrictive independence assumption it yields surprisingly good classifications [DPHS98, Joa98]. It may be extended into several directions [Seb02].

As the effort for manually labeling the documents of the training set is high, some authors use unlabeled documents for training. Assume that from a small training set it has been established that word t_i is highly correlated with class L_c. If from unlabeled documents it may be determined that word t_j is highly correlated with t_i, then also t_j is a good predictor for class L_c. In this way unlabeled documents may improve classification performance. In [NMTM00] the authors used a combination of Expectation-Maximization (EM) [DLR77] and a naïve Bayes classifier and were able to reduce the classification error by up to 30%.

3.1.3 Nearest Neighbor Classifier

Instead of building explicit models for the different classes we may select documents from the training set which are “similar” to the target document. The class of the target document subsequently may be inferred from the class labels of these similar documents. If k similar documents are considered, the approach is also known as k-*nearest neighbor classification*.

There is a large number of similarity measures used in text mining. One possibility is simply to count the number of common words in two documents. Obviously this has to be normalized to account for documents of different lengths. On the other hand words have greatly varying information content. A standard way to measure the latter is the cosine similarity as defined in (3). Note that only a small fraction of all possible terms appear in this sums as $w(d,t) = 0$ if the term t is not present in the document *d*. Other similarity measures are discussed in [BYRN99].

For deciding whether document d_i belongs to class L_m, the similarity $S(d_i, d_j)$ to all documents d_j in the training set is determined. The k most similar training documents (neighbors) are selected. The proportion of neighbors having the same class may be taken as an estimator for the probability of that class, and the class with the largest proportion is assigned to document d_i. The optimal number k of neighbors may be estimated from additional training data by cross-validation.

Nearest neighbor classification is a nonparametric method and it can be shown that for large data sets the error rate of the 1-nearest neighbor classifier is never larger than twice the optimal error rate [HTF01]. Several studies have shown that k-nearest neighbor methods have very good performance in practice [Joa98]. Their drawback is the computational effort during classification, where basically the similarity of a document with respect to all other documents of a training set has to be determined. Some extensions are discussed in [Seb02].

3.1.4 Decision Trees

Decision trees are classifiers which consist of a set of rules which are applied in a sequential way and finally yield a decision. They can be best explained by observing the training process, which starts with a comprehensive training set. It uses a divide and conquer strategy: For a training set M with labelled documents the word t_i is selected, which can predict the class of the documents in the best way, e.g. by the information gain (8). Then M is partitioned into two subsets, the subset M_i^+ with the documents containing t_i, and the subset M_i^- with the documents without t_i. This procedure is recursively applied to M_i^+ and M_i^-. It stops if all documents in a subset belong to the same class L_c. It generates a tree of rules with an assignment to actual classes in the leaves.

Decision trees are a standard tool in data mining [Qui86, Mit97]. They are fast and scalable both in the number of variables and the size of the training set. For text mining, however, they have the drawback that the final decision depends only on relatively few terms. A decisive improvement may be achieved by *boosting decision trees* [SS99], i.e. determining a set of complementary decision trees constructed in such a way that the overall error is reduced. [SS00] use even simpler one step decision trees containing only one rule and get impressive results for text classification.

3.1.5 Support Vector Machines and Kernel Methods

A Support Vector Machine (SVM) is a supervised classification algorithm that recently has been applied successfully to text classification tasks [Joa98, DPHS98, LK02]. As usual a document *d* is represented by a – possibly weighted – vector $(t_{d1}, \ldots, t_{dN})$ of the counts of its words. A single SVM can only separate two classes — a positive class L_1 (indicated by $y = +1$) and a negative class L_2 (indicated by $y = -1$). In the space of input vectors a hyperplane may be defined by setting $y = 0$ in the following linear

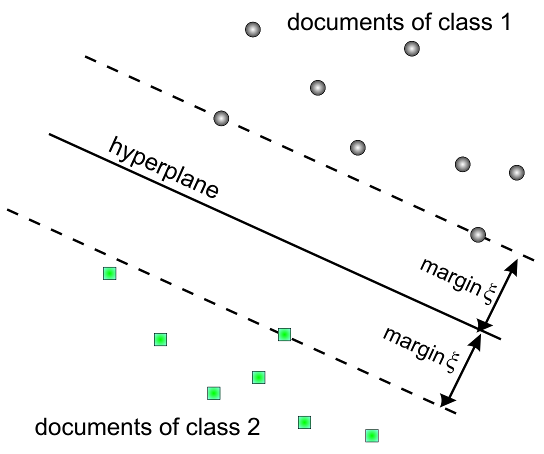


Figure 2: Hyperplane with maximal distance (margin) to examples of positive and negative classes constructed by the support vector machine.

equation.

$$y = f(\vec{t_d}) = b_0 + \sum_{j=1}^{N} b_j t_{dj}$$

The SVM algorithm determines a hyperplane which is located between the positive and negative examples of the training set. The parameters b_j are adapted in such a way that the distance ξ – called *margin* – between the hyperplane and the closest positive and negative example documents is maximized, as shown in Fig. 3.1.5. This amounts to a constrained quadratic optimization problem which can be solved efficiently for a large number of input vectors.

The documents having distance ξ from the hyperplane are called *support vectors* and determine the actual location of the hyperplane. Usually only a small fraction of documents are support vectors. A new document with term vector $\vec{t_d}$ is classified in L_1 if the value $f(\vec{t_d}) > 0$ and into L_2 otherwise. In case that the document vectors of the two classes are not linearly separable a hyperplane is selected such that as few as possible document vectors are located on the “wrong” side.

SVMs can be used with non-linear predictors by transforming the usual input features in a non-linear way, e.g. by defining a *feature map*

$$\phi(t_1, \ldots, t_N) = \left(t_1, \ldots, t_N, t_1^2, t_1 t_2, \ldots, t_N t_{N-1}, t_N^2\right)$$

Subsequently a hyperplane may be defined in the expanded input space. Obviously such non-linear transformations may be defined in a large number of ways.

The most important property of SVMs is that learning is nearly independent of the dimensionality of the feature space. It rarely requires feature selection as it inherently selects data points (the support vectors) required for a good classification. This allows good generalization even in the presence of a large number of features and makes SVM especially suitable for the classification of texts [Joa98]. In the case of textual data the choice of the kernel function has a minimal effect on the accuracy of classification: Kernels that imply a high dimensional feature space show slightly better results in terms of precision and recall, but they are subject to overfitting [LK02].

3.1.6 Classifier Evaluations

During the last years text classifiers have been evaluated on a number of benchmark document collections. It turns out that the level of performance of course depends on the document collection. Table 1 gives some representative results achieved for the Reuters 20 newsgroups collection [Seb02, p.38]. Concerning the relative quality of classifiers boosted trees, SVMs, and k-nearest neighbors usually deliver top-notch performance, while naïve Bayes and decision trees are less reliable.

Table 1: Performance of Different Classifiers for the Reuters collection

Method	F_1-value
naïve Bayes	0.795
decision tree C4.5	0.794
k-nearest neighbor	0.856
SVM	0.870
boosted tree	0.878

3.2 Clustering

Clustering method can be used in order to find groups of documents with similar content. The result of clustering is typically a partition (also called) clustering $\mathbb{P}$, a set of clusters P. Each cluster consists of a number of documents *d*. Objects — in our case documents — of a cluster should be similar and dissimilar to documents of other clusters. Usually the quality of clusterings is considered better if the contents of the documents within one cluster are more similar and between the clusters more dissimilar. Clustering methods group the documents only by considering their distribution in document space (for example, a n-dimensional space if we use the vector space model for text documents).

Clustering algorithms compute the clusters based on the attributes of the data and measures of (dis)similarity. However, the idea of what an ideal clustering result should look like varies between applications and might be even different between users. One can exert influence on the results of a clustering algorithm by using only subsets of attributes or by adapting the used similarity measures and thus control the clustering process. To which extent the result of the cluster algorithm coincides with the ideas of the user can be assessed by evaluation measures. A survey of different kinds of clustering algorithms and the resulting cluster types can be found in [SEK03].

In the following, we first introduce standard evaluation methods and present then details for hierarchical clustering approaches, k-means, bi-section-k-means, self-organizing maps and the EM-algorithm. We will finish the clustering section with a short overview of other clustering approaches used for text clustering.

3.2.1 Evaluation of clustering results

In general, there are two ways to evaluate clustering results. One the one hand statistical measures can be used to describe the properties of a clustering result. On the other hand some given classification can be seen as a kind of gold standard which is then typically used to compare the clustering results with the given classification. We discuss both aspects in the following.

Statistical Measures In the following, we first discuss measures which cannot make use of a given classification $\mathbb{L}$ of the documents. They are called indices in statistical literature and evaluate the quality of a clustering on the basis of statistic connections. One finds a large number of indices in literature (see [Fic97, DH73]). One of the most well-known measures is the mean square error. It permits to make statements on quality of the found clusters dependent on the number of clusters. Unfortunately, the computed quality is always better if the number of cluster is higher. In [KR90] an alternative measure, the silhouette coefficient, is presented which is independent of the number of clusters. We introduce both measures in the following.

Mean square error If one keeps the number of dimensions and the number of clusters constant the mean square error (Mean Square error, MSE) can be used likewise for the evaluation of the quality of clustering. The mean square error is a measure for the compactness of the clustering and is defined as follows:

Definition 1 (MSE) *The means square error (MSE) for a given clustering $\mathbb{P}$ is defined as*

$$MSE(\mathbb{P}) = \sum_{P \in \mathbb{P}} MSE(P), \tag{9}$$

whereas the means square error for a cluster P is given by:

$$MSE(P) = \sum_{d \in P} dist(d, \mu_P)^2, \tag{10}$$

and $\mu_P = \frac{1}{|P|} \sum_{d \in P} \vec{t_d}$ is the centroid of the clusters P and $dist$ is a distance measure.

Silhouette Coefficient One clustering measure that is independent from the number of clusters is the silhouette coefficient SC(P) (cf. [KR90]). The main idea of the coefficient is to find out the location of a document in the space with respect to the cluster of the document and the next similar cluster. For a good clustering the considered document is nearby the own cluster whereas for a bad clustering the document is closer to the next cluster. With the help of the silhouette coefficient one is able to judge the quality of a cluster or the entire clustering (details can be found in [KR90]). [KR90] gives characteristic values of the silhouette coefficient for the evaluation of the cluster quality. A value for SC(P) between 0.7 and 1.0 signals excellent separation between the found clusters, i.e. the objects within a cluster are very close to each other and are far away from other clusters. The structure was very well identified by the cluster algorithm. For the range from 0.5 to 0.7 the objects are clearly assigned to the appropriate clusters. A larger level of noise exists in the data set if the silhouette coefficient is within the range of 0.25 to 0.5 whereby also here still clusters are identifiable. Many objects could not be assigned clearly to one cluster in this case due to the cluster algorithm. At values under 0.25 it is practically impossible to identify a cluster structure and to calculate meaningful (from the view of application) cluster centers. The cluster algorithm more or less ”guessed” the clustering.

Comparative Measures The purity measure is based on the well-known precision measure for information retrieval (cf. [PL02]). Each resulting cluster P from a partitioning $\mathbb{P}$ of the overall document set D is treated as if it were the result of a query. Each set L of documents of a partitioning $\mathbb{L}$, which is obtained by manual labelling, is treated as if it is the desired set of documents for a query which leads to the same definitions for precision, recall and f-score as defined in Equations 6 and 7. The two partitions $\mathbb{P}$ and $\mathbb{L}$ are then compared as follows.

The precision of a cluster $P \in \mathbb{P}$ for a given category $L \in \mathbb{L}$ is given by

$$\text{Precision}(P, L) := \frac{|P \cap L|}{|P|}. \tag{11}$$

The overall value for purity is computed by taking the weighted average of maximal precision values:

$$\text{Purity}(\mathbb{P}, \mathbb{L}) := \sum_{P \in \mathbb{P}} \frac{|P|}{|D|} \max_{L \in \mathbb{L}} \text{Precision}(P, L). \tag{12}$$

The counterpart of purity is:

$$\text{InversePurity}(\mathbb{P}, \mathbb{L}) := \sum_{L \in \mathbb{L}} \frac{|L|}{|D|} \max_{P \in \mathbb{P}} \text{Recall}(P, L), \tag{13}$$

where $\text{Recall}(P, L) := \text{Precision}(L, P)$ and the well known

$$\text{F-Measure}(\mathbb{P}, \mathbb{L}) := \sum_{L \in \mathbb{L}} \frac{|L|}{|D|} \max_{P \in \mathbb{P}} \frac{2 \cdot \text{Recall}(P, L) \cdot \text{Precision}(P, L)}{\text{Recall}(P, L) + \text{Precision}(P, L)}, \tag{14}$$

which is based on the F-score as defined in Eq. 7.

The three measures return values in the interval $[0, 1]$, with 1 indicating optimal agreement. Purity measures the homogeneity of the resulting clusters when evaluated against a pre-categorization, while inverse purity measures how stable the pre-defined categories are when split up into clusters. Thus, purity achieves an “optimal” value of 1 when the number of clusters k equals $|D|$, whereas inverse purity achieves an “optimal” value of 1 when k equals 1. Another name in the literature for inverse purity is microaveraged precision. The reader may note that, in the evaluation of clustering results, microaveraged precision is identical to microaveraged recall (cf. e.g. [Seb02]). The F-measure works similar as inverse purity, but it depreciates overly large clusters, as it includes the individual precision of these clusters into the evaluation.

While (inverse) purity and F-measure only consider 'best' matches between 'queries' and manually defined categories, the *entropy* indicates how large the information content uncertainty of a clustering result with respect to the given classification is

$$E(\mathbb{P}, L) = \sum_{P \in \mathbb{P}} \text{prob}(P) \cdot E(P), \text{where} \tag{15}$$

$$E(P) = -\sum_{L \in \mathbb{L}} \text{prob}(L|P) \log(\text{prob}(L|P)) \tag{16}$$

where $\text{prob}(L|P) = \text{Precision}(P, L)$ and $\text{prob}(P) = \frac{|P|}{|D|}$. The entropy has the range [0, $log(|\mathbb{L}|)$], with 0 indicating optimality.

3.2.2 Partitional Clustering

Hierarchical Clustering Algorithms [MS01a, SKK00] got their name since they form a sequence of groupings or clusters that can be represented in a hierarchy of clusters. This hierarchy can be obtained either in a top-down or bottom-up fashion. Top-down means that we start with one cluster that contains all documents. This cluster is stepwise refined by splitting it iteratively into sub-clusters. One speaks in this case also of the so called "divisive" algorithm. The bottom-up or "agglomerative" procedures start by considering every document as individual cluster. Then the most similar clusters are iteratively merged, until all documents are contained in one single cluster. In practice the divisive procedure is almost of no importance due to its generally bad results. Therefore, only the agglomerative algorithm is outlined in the following.

The agglomerative procedure considers initially each document *d* of the the whole document set *D* as an individual cluster. It is the first cluster solution. It is assumed that each document is member of exactly one cluster. One determines the similarity between the clusters on the basis of this first clustering and selects the two clusters p, q of the clustering $\mathbb{P}$ with the minimum distance $dist(p, q)$. Both cluster are merged and one receives a new clustering. One continues this procedure and re-calculates the distances between the new clusters in order to join again the two clusters with the minimum distance $dist(p, q)$. The algorithm stops if only one cluster is remaining.

The distance can be computed according to Eq. 4. It is also possible to derive the clusters directly on the basis of the similarity relationship given by a matrix. For the computation of the similarity between clusters that contain more than one element different distance measures for clusters can be used, e.g. based on the outer cluster shape or the cluster center. Common linkage procedures that make use of different cluster distance measures are single linkage, average linkage or Ward's procedure. The obtained clustering depends on the used measure. Details can be found, for example, in [DH73].

By means of so-called dendrograms one can represent the hierarchy of the clusters obtained as a result of the repeated merging of clusters as described above. The dendrograms allows to estimate the number of clusters based on the distances of the merged clusters. Unfortunately, the selection of the appropriate linkage method depends on the desired cluster structure, which is usually unknown in advance. For example, single linkage tends to follow chain-like clusters in the data, while complete linkage tends to create ellipsoid clusters. Thus prior knowledge about the expected distribution and cluster form is usually necessary for the selection of the appropriate method (see also [DH73]). However, substantially more problematic for the use of the algorithm for large data sets is the memory required to store the similarity matrix, which consists of $n(n-1)/2$ elements where n is the number of documents. Also the runtime behavior with $O(n^2)$ is worse compared to the linear behavior of k-means as discussed in the following.

k-means is one of the most frequently used clustering algorithms in practice in the field of data mining and statistics (see [DH73, Har75]). The procedure which originally comes from statistics is simple to implement and can also be applied to large data sets. It turned out that especially in the field of text clustering k-means obtains good results. Proceeding from a starting solution in which all documents are distributed on a given number of clusters one tries to improve the solution by a specific change of the allocation of documents to the clusters. Meanwhile, a set of variants exists whereas the basic principle goes back to Forgy 1965 [For65] or MacQueen 1967 [Mac67]. In literature for vector quantization k-means is also known under the name LloydMaxAlgorithm ([GG92]). The basic principle is shown in the following algorithm:

Algorithm 1 The k-means algorithm

Input: set *D*, distance measure $dist$, number k of cluster
Output: A partitioning $\mathbb{P}$ of the set *D* of documents (i. e., a set $\mathbb{P}$ of k disjoint subsets of *D* with $\bigcup_{P\in\mathbb{P}} P =$ *D*).

```
1: Choose randomly $k$ data points from D as starting centroids $\vec{t_{P_1}} \dots \vec{t_{P_k}}$.
2: repeat
3:     Assign each point of $P$ to the closest centroid with respect to $dist$.
4:     (Re-)calculate the cluster centroids $\vec{t_{P_1}} \dots \vec{t_{P_k}}$ of clusters $P_1 \dots P_k$.
5: until cluster centroids $\vec{t_{P_1}} \dots \vec{t_{P_k}}$ are stable
6: return set $\mathbb{P} := \{P_1, \dots, P_k\}$, of clusters.
```

k-means essentially consists of the steps three and four in the algorithm, whereby the number of clusters k must be given. In step three the documents are assigned to the nearest of the k centroids (also called cluster *prototype*). Step four calculates a new centroids on the basis of the new allocations. We repeat the two steps in a loop (step five) until the cluster centroids do not change any more. The algorithm 5.1 corresponds to a simple hill climbing procedure which typically gets stuck in a local optimum (the finding of the global optimum is a NP complete problem). Apart from a suitable method to determine the starting solution (step one), we require a measure for calculating the distance or similarity in step three (cf. section 2.1). Furthermore the abort criterion of the loop in step five can be chosen differently e.g. by stopping after a fix number of iterations.

Bi-Section-k-means One fast text clustering algorithm, which is also able to deal with the large size of the textual data is the Bi-Section-k-means algorithm. In [SKK00] it was shown that Bi-Section-k-means is a fast and high-quality clustering algorithm for text documents which is frequently outperforming standard k-means as well as agglomerative clustering techniques.

Bi-Section-k-means is based on the k-means algorithm. It repeatedly splits the largest cluster (using k-means) until the desired number of clusters is obtained. Another way of choosing the next cluster to be split is picking the one with the largest variance. [SKK00] showed neither of these two has a significant advantage.

Self Organizing Map (SOM) [Koh82] are a special architecture of neural networks that cluster high-dimensional data vectors according to a similarity measure. The clusters are arranged in a low-dimensional topology that preserves the neighborhood relations in the high dimensional data. Thus, not only objects that are assigned to one cluster are similar to each other (as in every cluster analysis), but also objects of nearby clusters are expected to be more similar than objects in more distant clusters. Usually, two-dimensional grids of squares or hexagons are used (cf. Fig. 3).

The network structure of a self-organizing map has two layers (see Fig. 3). The neurons in the input layer correspond to the input dimensions, here the words of the document vector. The output layer (map) contains as many neurons as clusters needed. All neurons in the input layer are connected with all neurons in the output layer. The weights of the connection between input and output layer of the neural network encode positions in the high-dimensional data space (similar to the cluster prototypes in k-means). Thus, every unit in the output layer represents a cluster center. Before the learning phase of the network, the two-dimensional structure of the output units is fixed and the weights are initialized randomly. During learning, the sample vectors (defining the documents) are repeatedly propagated through the network. The weights of the most similar prototype $\vec{w_s}$ (*winner neuron*) are modified such that the prototype moves toward the input vector $\vec{w_i}$, which is defined by the currently considered document d, i.e. $\vec{w_i} := \vec{t_d}$ (*competitive learning*). As similarity measure usually the Euclidean distance is used. However, for text documents the scalar product (see Eq. 3) can be applied. The weights $\vec{w_s}$ of the winner neuron are modified according to the following equation:

$$\vec{w_s}' = \vec{w_s} + \sigma \cdot (\vec{w_s} - \vec{w_i}),$$

where σ is a learning rate.

To preserve the neighborhood relations, prototypes that are close to the winner neuron in the two-dimensional structure are also moved in the same direction. The weight change decreases with the distance from the winner neuron. Therefore, the adaption method is extended by a neighborhood function v (see also Fig. 3):

$$\vec{w_s}' = \vec{w_s} + v(i, s) \cdot \sigma \cdot (\vec{w_s} - \vec{w_i}),$$

where σ is a learning rate. By this learning procedure, the structure in the high-dimensional sample data is non-linearly projected to the lower-dimensional topology. After learning, arbitrary vectors (i.e. vectors from the sample set or prior ‘unknown’ vectors) can be propagated through the network and are mapped to the output units.

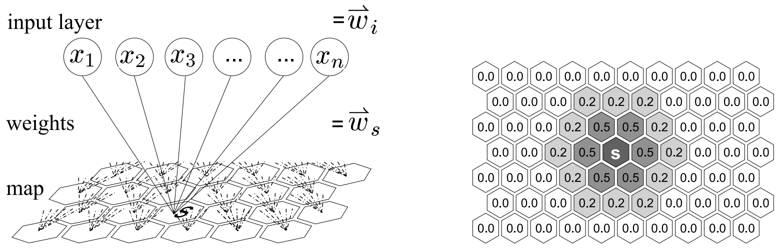


Figure 3: Network architecture of self-organizing maps (*left*) and possible neighborhood function v for increasing distances from s (*right*)

For further details on self-organizing maps see [Koh84]. Examples for the application of SOMs for text mining can be found in [LMS91, HKLK96, KKL$^+$00, Nür01, RC01] and in Sect. 3.4.2.

Model-based Clustering Using the EM-Algorithm Clustering can also be viewed from a statistical point of view. If we have k different clusters we may either assign a document $\boldsymbol{d}_i$ with certainty to a cluster (hard clustering) or assign $\boldsymbol{d}_i$ with probability q_{ic} to P_c (soft clustering), where $q_i = (q_{i1}, \ldots, q_{ik})$ is a probability vector $\sum_{c=1}^{k} q_{ic} = 1$.

The underlying statistical assumption is that a document was created in two stages: First we pick a cluster P_c from $\{1, \ldots, k\}$ with fixed probability q_c; then we generate the words t of the document according to a cluster-specific probability distribution $p(t|P_c)$. This corresponds to a mixture model where the probability of an observed document $(t_1, \ldots, t_{n_i})$ is

$$p(t_1, \ldots, t_{n_i}) = \sum_{c=1}^{k} q_c p(t_1, \ldots, t_{n_i}|P_c) \tag{17}$$

Each cluster P_c is a mixture component. The mixture probabilities q_c describe an unobservable "cluster variable" z which may take the values from $\{1, \ldots, k\}$. A well established method for estimating models involving unobserved variables is the EM-algorithm [HTF01], which basically replaces the unknown value with its current probability estimate and then proceeds as if it has been observed. Clustering methods for documents based on mixture models have been proposed by Cheeseman [CS96] and yield excellent results. Hofmann [Hof01] formulates a variant that is able to cluster terms occurring together instead of documents.

3.2.3 Alternative Clustering Approaches

Co-clustering algorithm designate the simultaneous clustering of documents and terms [DMM03]. They follow thereby another paradigm than the ”classical” cluster algorithm as k-means which only clusters elements of the one dimension on the basis of their similarity to the second one, e.g. documents based on terms.

Fuzzy Clustering While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for *degrees of membership*, to which a datum belongs to different clusters [Bez81]. These approaches are frequently more stable. Applications to text are described in, e.g., [MS01b, BN04].

The Utility of Clustering We have described the most important types of clustering approaches, but we had to leave out many other. Obviously there are many ways to define clusters and because of this we cannot expect to obtain something like the 'true' clustering. Still clustering can be insightful. In contrast to classification, which relies on a prespecified grouping, cluster procedures label documents in a new way. By studying the words and phrases that characterize a cluster, for example, a company could learn new insights about its customers and their typical properties. A comparison of some clustering methods is given in [SKK00].

3.3 Information Extraction

Natural language text contains much information that is not directly suitable for automatic analysis by a computer. However, computers can be used to sift through large amounts of text and extract useful information from single words, phrases or passages. Therefore *information extraction* can be regarded as a restricted form of full natural language understanding, where we know in advance what kind of semantic information we are looking for. The main task is to extract parts of text and assign specific attributes to it.

As an example consider the task to extract executive position changes from news stories: "Robert L. James, chairman and chief executive officer of McCann-Erickson, is going to retire on July 1st. He will be replaced by John J. Donner, Jr., the agencies chief operating officer." In this case we have to identify the following information: Organization (McCann-Erickson), position (chief executive officer), date (July 1), outgoing person name (Robert L. James), and incoming person name (John J. Donner, Jr.).

The task of information extraction naturally decomposes into a series of processing steps, typically including tokenization, sentence segmentation, part-of-speech assignment, and the identification of named entities, i.e. person names, location names and names of organizations. At a higher level phrases and sentences have to be parsed, semantically interpreted and integrated. Finally the required pieces of information like "position" and "incoming person name" are entered into the database. Although the most accurate information extraction systems often involve handcrafted language-processing modules, substantial progress has been made in applying data mining techniques to a number of these steps.

3.3.1 Classification for Information Extraction

Entity extraction was originally formulated in the Message Understanding Conference [Chi97]. One can regard it as a word-based tagging problem: The word, where the entity starts, get tag "B", continuation words get tag "I" and words outside the entity get tag ”O”. This is done for each type of entity of interest. For the example above we have for instance the person-words ”by (O) John (B) J. (I) Donner (I) Jr. (I) the (O)”.

Hence we have a sequential classification problem for the labels of each word, with the surrounding words as input feature vector. A frequent way of forming the feature vector is a binary encoding scheme. Each feature component can be considered as a test that asserts whether a certain pattern occurs at a specific position or not. For example, a feature component takes the value 1 if the previous word is the word ”John” and 0 otherwise. Of course we may not only test the presence of specific words but also whether the words starts with a capital letter, has a specific suffix or is a specific part-of-speech. In this way results of previous analysis may be used.

Now we may employ any efficient classification method to classify the word labels using the input feature vector. A good candidate is the Support Vector Machine because of its ability to handle large sparse feature vectors efficiently. [TC02] used it to extract entities in the molecular biology domain.

3.3.2 Hidden Markov Models

One problem of standard classification approaches is that they do not take into account the predicted labels of the surrounding words. This can be done using probabilistic models of sequences of labels and features. Frequently used is the hidden Markov model (HMM), which is based on the conditional distributions of current labels $L^{(j)}$ given the previous label $L^{(j-1)}$ and the distribution of the current word $t^{(j)}$ given the current and the previous labels $L^{(j)}, L^{(j-1)}$.

$$L^{(j)} \sim p(L^{(j)}|L^{(j-1)}) \qquad t^{(j)} \sim p(t^{(j)}|L^{(j)}, L^{(j-1)}) \tag{18}$$

A training set of words and their correct labels is required. For the observed words the algorithm takes into account all possible sequences of labels and computes their probabilities. An efficient learning method that exploits the sequential structure is the Viterbi algorithm [Rab89]. Hidden Markov models were successfully used for named entity extraction, e.g. in the Identifinder system [BSW99].

3.3.3 Conditional Random Fields

Hidden Markov models require the conditional independence of features of different words given the labels. This is quite restrictive as we would like to include features which correspond to several words simultaneously. A recent approach for modelling this type of data is called *conditional random field* (CRF) [LMP01]. Again we consider the observed vector of words $\mathbf{t}$ and the corresponding vector of labels $\mathbf{L}$. The labels have a graph structure. For a label L_c let $N(c)$ be the indices of neighboring labels. Then $(\mathbf{t}, \mathbf{L})$ is a conditional random field when conditioned on the vector $\mathbf{t}$ of all terms the random variables obey the Markov property

$$p(L_c|\mathbf{t}, L_d; d \neq c) = p(L_c|\mathbf{t}, L_d; d \in N(c)) \tag{19}$$

i.e. the whole vector $\mathbf{t}$ of observed terms and the labels of neighbors may influence the distribution of the label L_c. Note that we do not model the distribution $p(\mathbf{t})$ of the observed words, which may exhibit arbitrary dependencies.

We consider the simple case that the words $\mathbf{t} = (t_1, t_2, \ldots, t_n)$ and the corresponding labels $L_1, L_2, \ldots, L_n$ have a chain structure and that L_c depends only on the preceding and succeeding labels L_{c-1} and L_{c+1}. Then the conditional distribution $p(\mathbf{L}|\mathbf{t})$ has the form

$$p(\mathbf{L}|\mathbf{t}) = \frac{1}{\text{const}} \exp\left(\sum_{j=1}^{n} \sum_{r=1}^{k_j} \lambda_{jr} f_{jr}(L_j, \mathbf{t}) + \sum_{j=1}^{n-1} \sum_{r=1}^{m_j} \mu_{jr} g_{jr}(L_j, L_{j-1}, \mathbf{t}) \right) \tag{20}$$

where $f_{jr}(L_j, \mathbf{t})$ and $g_{jr}(L_j, L_{j-1}, \mathbf{t})$ are different features functions related to L_j and the pair L_j, L_{j-1} respectively. CRF models encompass hidden Markov models, but they are much more expressive because they allow arbitrary dependencies in the observation sequence and more complex neighborhood structures of labels. As for most machine learning algorithms a training sample of words and the correct labels is required. In addition to the identity of words arbitrary properties of the words, like part-of-speech tags, capitalization, prefixes and suffixes, etc. may be used leading to sometimes more than a million features. The unknown parameter values λ_{jr} and μ_{jr} are usually estimated using conjugate gradient optimization routines [McC03].

McCallum [McC03] applies CRFs with feature selection to named entity recognition and reports the following F1-measures for the CoNLL corpus: person names 93%, location names 92%, organization names 84%, miscellaneous names 80%. CRFs also have been successfully applied to noun phrase identification [McC03], part-of-speech tagging [LMP01], shallow parsing [SP03], and biological entity recognition [KOT$^+$04].

3.4 Explorative Text Mining: Visualization Methods

Graphical visualization of information frequently provides more comprehensive and better and faster understandable information than it is possible by pure text based descriptions and thus helps to mine large document collections. Many of the approaches developed for text mining purposes are motivated by methods that had been proposed in the areas of explorative data analysis, information visualization and visual data mining. For an overview of these areas of research see, e.g., [UF01, Kei02]. In the following we will focus on methods that have been specifically designed for text mining or — as a subgroup of text mining methods and a typical application of visualization methods — information retrieval.

In text mining or information retrieval systems visualization methods can improve and simplify the discovery or extraction of relevant patterns or information. Information that allow a visual representation comprises aspects of the document collection or result sets, keyword relations, ontologies or — if retrieval systems are considered — aspects of the search process itself, e.g. the search or navigation path in hyperlinked collections.

However, especially for text collections we have the problem of finding an appropriate visualization for abstract textual information. Furthermore, an *interactive* visual data exploration interface is usually desirable, e.g. to zoom in local areas or to select or mark parts for further processing. This results in great demands on the user interface and the hardware. In the following we give a brief overview of visualization methods that have been realized for text mining and information retrieval systems.

3.4.1 Visualizing Relations and Result Sets

Interesting approaches to visualize keyword-document relations are, e.g., the Cat-a-Cone model [HK97], which visualizes in a three dimensional representation hierarchies of categories that can be interactively used to refine a search. The InfoCrystal [Spo95] visualizes a (weighted) boolean query and the belonging result set in a crystal structure. The lyberworld model [HKW94] and the visualization components of the SENTINEL Model [FFKS99] are representing documents in an abstract keyword space.

An approach to visualize the results of a set of queries was presented in [HHP$^+$01]. Here, retrieved documents are arranged according to their similarity to a query on straight lines. These lines are arranged in a circle around a common center, i.e. every query is represented by a single line. If several documents are placed on the same (discrete) position, they are arranged in the same distance to the circle, but with a slight offset. Thus, clusters occur that represent the distribution of documents for the belonging query.

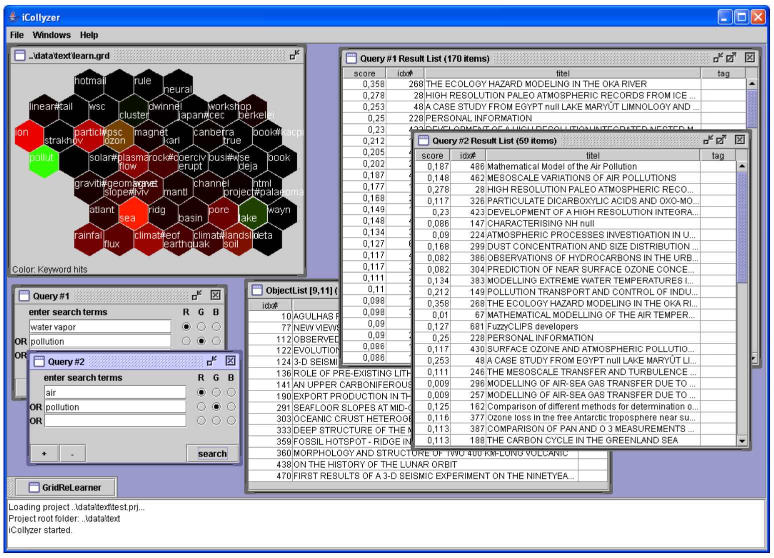
3.4.2 Visualizing Document Collections

For the visualization of document collections usually two-dimensional projections are used, i.e. the high dimensional document space is mapped on a two-dimensional surface. In order to depict individual documents or groups of documents usually text flags are used, which represent either a keyword or the document category. Colors are frequently used to visualize the density, e.g. the number of documents in this area, or the difference to neighboring documents, e.g. in order to emphasize borders between different categories. If three-dimensional projections are used, for example, the number of documents assigned to a specific area can be represented by the z-coordinate.

An Example: Visualization using Self-Organizing Maps Visualization of document collections requires methods that are able to group documents based on their similarity and furthermore that visualize the similarity between discovered groups of documents. Clustering approaches that are frequently used to find groups of documents with similar content [SKK00] – see also section 3.2 – usually do not consider the neighborhood relations between the obtained cluster centers. Self-organizing maps, as discussed above, are an alternative approach which is frequently used in data analysis to cluster high dimensional data. The resulting clusters are arranged in a low-dimensional topology that preserves the neighborhood relations of the corresponding high dimensional data vectors and thus not only objects that are assigned to one cluster are similar to each other, but also objects of nearby clusters are expected to be more similar than objects in more distant clusters.

Usually, two-dimensional arrangements of squares or hexagons are used for the definition of the neighborhood relations. Although other topologies are possible for self-organizing maps, two-dimensional maps have the advantage of intuitive visualization and thus good exploration possibilities. In document retrieval, self-organizing maps can be used to arrange documents based on their similarity. This approach opens up several appealing navigation possibilities. Most important, the surrounding grid cells of documents known to be interesting can be scanned for further similar documents. Furthermore, the distribution of keyword search results can be visualized by coloring the grid cells of the map with respect to the number of hits. This allows a user to judge e.g. whether the search results are assigned to a small number of (neighboring) grid cells of the map, or whether the search hits are spread widely over the map and thus the search was - most likely - too unspecific.

A first application of self-organizing maps in information retrieval was presented in [LMS91]. It provided a simple two-dimensional cluster representation (categorization) of a small document collection. A refined model, the WEBSOM approach, extended this idea to a web based interface applied to newsgroup data that provides simple zooming techniques and coloring methods [HKLK96, Hon97, KKL$^+$00]. Further extensions introduced hierarchies [Mer98], supported the visualization of search results [RC01] and combined search, navigation and visualization techniques in an integrated tool [Nür01]. A screenshot of the prototype discussed in [Nür01] is depicted in Fig. 4.



Query #1 Result List (170 items)

score	idx#	titel	tag
0,358	268	THE ECOLOGY HAZARD MODELING IN THE OKA RIVER	
0,278	28	HIGH RESOLUTION PALEO ATMOSPHERIC RECORDS FROM ICE ...	
0,253	48	A CASE STUDY FROM EGYPT null LAKE MARYÛT LIMNOLOGY AND ...	
0,25	228	PERSONAL INFORMATION	
0,23			
0,212			
0,205			
0,202			
0,187			
0,177			
0,168			
0,149			
0,148			
0,134			
0,127			
0,117			
0,117			
0,111			
0,11			
0,098			
0,098			
0,09			
0,09			
0,086			
0,086			

Query #2 Result List (59 items)

score	idx#	titel	tag
0,187	486	Mathematical Model of the Air Pollution	
0,148	462	MESOSCALE VARIATIONS OF AIR POLLUTIONS	
0,278	28	HIGH RESOLUTION PALEO ATMOSPHERIC RECO...	
0,117	326	PARTICULATE DICARBOXYLIC ACIDS AND OXO-MO...	
0,23	423	DEVELOPMENT OF A HIGH RESOLUTION INTEGRA...	
0,086	147	CHARACTERISING NH null	
0,09	224	ATMOSPHERIC PROCESSES INVESTIGATION IN U...	
0,168	299	DUST CONCENTRATION AND SIZE DISTRIBUTION ...	
0,082	386	OBSERVATIONS OF HYDROCARBONS IN THE URB...	
0,082	304	PREDICTION OF NEAR SURFACE OZONE CONCE...	
0,134	383	MODELLING EXTREME WATER TEMPERATURES I...	
0,212	149	POLLUTION TRANSPORT AND CONTROL OF INDU...	
0,358	268	THE ECOLOGY HAZARD MODELING IN THE OKA RI...	
0,01	67	MATHEMATICAL MODELLING OF THE AIR TEMPER...	
0,127	681	FuzzyCLIPS developers	
0,25	228	PERSONAL INFORMATION	
0,117	430	SURFACE OZONE AND ATMOSPHERIC POLLUTIO...	
0,253	48	A CASE STUDY FROM EGYPT null LAKE MARYÛT LI...	
0,111	246	THE MESOSCALE TRANSFER AND TURBULENCE ...	
0,009	296	MODELLING OF AIR-SEA GAS TRANSFER DUE TO ...	
0,009	257	MODELLING OF AIR-SEA GAS TRANSFER DUE TO ...	
0,125	162	Comparison of different methods for determination o...	
0,116	377	Ozone loss in the free Antarctic troposphere near su...	
0,113	387	COMPARISON OF PAN AND O 3 MEASUREMENTS ...	
0,113	188	THE CARBON CYCLE IN THE GREENLAND SEA	

ObjectList [9,11] (

idx#	
10	AGULHAS
77	NEW VIEWS
112	OBSERVED
122	EVOLUTION
124	3-D SEISMI
136	ROLE OF PRE-EXISTING LITH
141	AN UPPER CARBONIFEROUS
190	EXPORT PRODUCTION IN TH
291	SEAFLOOR SLOPES AT MID-
303	OCEANIC CRUST HETEROGE
333	DEEP STRUCTURE OF THE
359	FOSSIL HOTSPOT - RIDGE IN
360	MORPHOLOGY AND STRUCTURE OF TWO 400 KM-LONG VOLCANIC
438	ON THE HISTORY OF THE LUNAR ORBIT
470	FIRST RESULTS OF A 3-D SEISMIC EXPERIMENT ON THE NINETYEA...

Query #1

	enter search terms	R	G	B
	water vapor	●	○	○
OR	pollution	○	●	○
OR				

Query #2

	enter search terms	R	G	B
	air	●	○	○
OR	pollution	○	●	○
OR		○	○	○

+ - search

```
Loading project ..\data\text\test.prj...
Project root folder: ..\data\text
iCollyzer started.
```

Figure 4: A Prototypical Retrieval System Based on Self-Organizing Maps

3.4.3 Other Techniques

Besides methods based on self-organizing maps several other techniques have been successfully applied to visualize document collections. For example, the tool VxInsight [BWD02] realizes a partially interactive mapping by an energy minimization approach similar to simulated annealing to construct a three dimensional landscape of the document collection. As input either a vector space description of the documents or a list of directional edges, e.g. defined based on citations of links, can be used. The tool

SPIRE [WTP$^+$95] applies a three step approach: It first clusters documents in document space, than projects the discovered cluster centers onto a two dimensional surface and finally maps the documents relative to the projected cluster centers. SPIRE offers a scatter plot like projection as well as a three dimensional visualization. The visualization tool SCI-Map [Sma99] applies an iterative clustering approach to create a network using, e.g., references of scientific publications. The tools visualizes the structure by a map hierarchy with an increasing number of details.

One major problem of most existing visualization approaches is that they create their output only by use of data inherent information, i.e. the distribution of the documents in document space. User specific information can not be integrated in order to obtain, e.g., an improved separation of the documents with respect to user defined criteria like keywords or phrases. Furthermore, the possibilities for a user to interact with the system in order to navigate or search are usually very limited, e.g., to boolean keyword searches and simple result lists.

3.5 Further Application Areas

Further major applications of text mining methods consider the detection of topics in text streams and text summarization.

Topic detection studies the problem of detecting new and upcoming topics in time-ordered document collections. The methods are frequently used in order to detect and monitor (*topic tracking*) news tickers or news broadcasts. An introduction and overview of current approaches can be found in [All02].

Text summarization aims at the creation of a condensed version of a document or a document collection (multidocument summarization) that should contain its most important topics. Most approaches still focus on the idea to extract individual informative sentences from a text. The summary consists then simply of a collection of these sentences. However, recently refined approaches try to extract semantic information from documents and create summaries based on this information (cf. [LGMF04]). For an overview see [MM99] and [RHM02].

4 Applications

In this section we briefly discuss successful applications of text mining methods in quite diverse areas as patent analysis, text classification in news agencies, bioinformatics and spam filtering. Each of the applications has specific characteristics that had to be considered while selecting appropriate text mining methods.

4.1 Patent Analysis

In recent years the analysis of patents developed to a large application area. The reasons for this are on the one hand the increased number of patent applications and on the other hand the progress that had been made in text classification, which allows to use these techniques in this due to the commercial impact quite sensitive area. Meanwhile, supervised and unsupervised techniques are applied to analyze patent documents and to support companies and also the European patent office in their work. The challenges in patent analysis consists of the length of the documents, which are larger then documents usually used in text classification, and the large number of available documents in a corpus [KSB01]. Usually every document consist of 5000 words in average. More than 140000 documents have to be handled by the European patent office (EPO) per year. They are processed by 2500 patent examiners in three locations.

In several studies the classification quality of state-of-the-art methods was analyzed. [KSB01] reported very good result with an 3% error rate for 16000 full text documents to be classified in 16 classes (mono-classification) and a 6% error rate in the same setting for abstracts only by using the Winnow [Lit88] and the Rocchio algorithm [Roc71]. These results are possible due to the large amount of available training documents. Good results are also reported in [KZ02] for an internal EPO text classification application with a precision of 81 % and an recall of 78 %.

Text clustering techniques for patent analysis are often applied to support the analysis of patents in large companies by structuring and visualizing the investigated corpus. Thus, these methods find their way in a lot of commercial products but are still also of interest for research, since there is still a need for improved performance. Companies like IBM offer products to support the analysis of patent text documents. Dorre describes in [DGS99] the IBM Intelligent Miner for text in a scenario applied to patent text and compares it also to data mining and text mining. Coupet [CH98] does not only apply clustering but also gives some nice visualization. A similar scenario on the basis of SOM is given in [LAHF03].

4.2 Text Classification for News Agencies

In publishing houses a large number of news stories arrive each day. The users like to have these stories tagged with categories and the names of important persons, organizations and places. To automate this process the Deutsche Presse-Agentur (dpa) and a group of leading German broadcasters (PAN) wanted to select a commercial text classification system to support the annotation of news articles. Seven systems were tested with a two given test corpora of about half a million news stories and different categorical hierarchies of about 800 and 2300 categories [Pd05]. Due to confidentiality the results can be published only in anonymized form.

For the corpus with 2300 categories the best system achieved at an F1-value of 39%, while for the corpus with 800 categories an F1-value of 79% was reached. In the latter case a partially automatic assignment based on the reliability score was possible for about half the documents, while otherwise the systems could only deliver proposals for human categorizers. Especially good are the results for recovering persons and geographic locations with about 80% F1-value. In general there were great variations between the performances of the systems.

In a usability experiment with human annotators the formal evaluation results were confirmed leading to faster and more consistent annotation. It turned out, that with respect to categories the human annotators exhibit a relative large disagreement and a lower consistency than text mining systems. Hence the support of human annotators by text mining systems offers more consistent annotations in addition to faster annotation.

The Deutsche Presse-Agentur now is routinely using a text mining system in its news production workflow.

4.3 Bioinformatics

Bio-entity recognition aims to identify and classify technical terms in the domain of molecular biology that correspond to instances of concepts that are of interest to biologists. Examples of such entities include the names of proteins, genes and their locations of activity such as cells or organism names. Entity recognition is becoming increasingly important with the massive increase in reported results due to high throughput experimental methods. It can be used in several higher level information access tasks such as relation extraction, summarization and question answering.

Recently the GENIA corpus was provided as a benchmark data set to compare different entity extraction approaches [KOT+04]. It contains 2000 abstracts from the MEDLINE database which were hand annotated with 36 types of biological entities. The following sentence is an example: "We have shown that <*protein*> interleukin-1 </*protein*> (<*protein*> IL-1 </*protein*>) and <*protein*> IL-2 </*protein*> control <*DNA*> IL-2 receptor alpha (IL-2R alpha) gene </*DNA*> transcription in <*cell_line*> CD4-CD8- murine T lymphocyte precursors </*cell_line*>".

In the 2004 evaluation four types of extraction models were used: Support Vector Machines (SVMs), Hidden Markov Models (HMMs), Conditional Random Fields (CRFs) and the related Maximum Entropy Markov Models (MEMMs). Varying types of input features were employed: lexical features (words), n-grams, orthographic information, word lists, part-of-speech tags, noun phrase tags, etc. The evaluation shows that the best five systems yield an F1-value of about 70% [KOT+04]. They use SVMs in combination with Markov models (72.6%), MEMMs (70.1%), CRFs (69.8%), CRFs together with SVMs (66.3%), and HMMs (64.8%). For practical applications the current accuracy levels are not yet satisfactory and research currently aims at including a sophisticated mix of external resources such as keyword lists and ontologies which provide terminological resources.

4.4 Anti-Spam Filtering of Emails

The explosive growth of unsolicited e-mail, more commonly known as spam, over the last years has been undermining constantly the usability of e-mail. One solution is offered by anti-spam filters. Most commercially available filters use black-lists and hand-crafted rules. On the other hand, the success of machine learning methods in text classification offers the possibility to arrive at anti-spam filters that quickly may be adapted to new types of spam.

There is a growing number of learning spam filters mostly using naive Bayes classifiers. A prominent example is Mozilla's e-mail client. Michelakis et al. [MAP+04] compare different classifier methods and investigate different costs of classifying a proper mail as spam. They find that for their benchmark corpora the SVM nearly always yields best results.

To explore how well a learning-based filter performs in real life, they used an SVM-based procedure for seven months without retraining. They achieved a precision of

96.5% and a recall of 89.3%. They conclude that these good results may be improved by careful preprocessing and the extension of filtering to different languages.

5 Conclusion

In this article, we tried to give a brief introduction to the broad field of text mining. Therefore, we motivated this field of research, gave a more formal definition of the terms used herein and presented a brief overview of currently available text mining methods, their properties and their application to specific problems. Even though, it was impossible to describe all algorithms and applications in detail within the (size) limits of an article, we think that the ideas discussed and the provided references should give the interested reader a rough overview of this field and several starting points for further studies.

References

[Abn91] S. P. Abney. Parsing by chunks. In R. C. Berwick, S. P. Abney, and C. Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer Academic Publishers, Boston, 1991.

[All02] J. Allan, editor. *Topic Detection and Tracking*. Kluwer Academic Publishers, Norwell, MA, 2002.

[Be99] M. B. and D. J. Hand (eds.). *Intelligent data analysis*. Springer-Verlag New York, Inc., 1999.

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[BH04] S. Bloehdorn and A. Hotho. Text classification by boosting weak learners based on terms and concepts. In *Proc. IEEE Int. Conf. on Data Mining (ICDM 04)*, pages 331–334. IEEE Computer Society Press, NOV 2004.

[BN04] C. Borgelt and A. Nürnberger. Fast fuzzy clustering of web page collections. In *Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM)*, Pisa, Italy, 2004.

[BSW99] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34:211–231, 1999.

[BWD02] K. W. Boyack, B. N. Wylie, and G. S. Davidson. Domain visualization using vxinsight for science and technology management. *Journal of the American Society for Information Science and Technologie*, 53(9):764–774, 2002.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley Longman, 1999.

[CH98] P. Coupet and M. Hehenberger. Text mining applied to patent analysis. In *Annual Meeting of American Intellectual Property Law Association (AIPLA) Airlington*, 1998.

[Chi97] N. Chinchor. Muc-7 named entity task definition version 3.5. Technical report, NIST, ftp.muc.saic.com/pub/MUC/MUC7-guidelines, 1997.

[CHY96] M.-S. Chen, J. Han, and P. S. Yu. Data mining: an overview from a database perspective. *IEEE Transaction on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[cri99] Cross industry standard process for data mining. `http://www.crisp-dm.org/`, 1999.

[CS96] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[DDFL90] S. Deerwester, S.T. Dumais, G.W. Furnas, and T.K. Landauer. Indexing by latent semantic analysis. *Journal of the American Society for Information Sciences*, 41:391–407, 1990.

[DGS99] J. Dörre, P. Gerstl, and R. Seiffert. Text mining: finding nuggets in mountains of textual data. In *Proc. 5th ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD-99)*, pages 398–401, San Diego, US, 1999. ACM Press, New York, US.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. J. Wiley & Sons, New York, NY, USA, 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistic Society, Series B*, 39(1):1–38, 1977.

[DMM03] I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *Proc. of the ninth ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining*, pages 89–98. ACM Press, 2003.

[DPHS98] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th Int. Conf. on Information and Knowledge Managment*, 1998.

[FBY92] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, New Jersey, 1992.

[FD95] R. Feldman and I. Dagan. Kdt - knowledge discovery in texts. In *Proc. of the First Int. Conf. on Knowledge Discovery (KDD)*, pages 112–117, 1995.

[FFKS99] K. L. Fox, O. Frieder, M. M. Knepper, and E. J. Snowberg. Sentinel: A multiple engine information retrieval and visualization system. *Journal of the American Society of Information Science*, 50(7):616–625, 1999.

[Fic97] N. Fickel. Clusteranalyse mit gemischt-skalierten merkmalen: Abstrahierung vom skalenniveau. *Allg. Statistisches Archiv*, 81(3):249–265, 1997.

[For65] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–769, 1965.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[Gai03] R. Gaizauskas. An information extraction perspective on text mining: Tasks, technologies and prototype applications. `http://www.itri.bton.ac.uk/projects/euromap/TextMiningEvent/Rob_Gaizauskas.pdf`, 2003.

[GG92] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[Goo65] I. J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press, Cambridge, MA, 1965.

[Gre98] W. R. Greiff. A theory of term weighting based on exploratory data analysis. In *21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, 1998. ACM.

[Har75] J. Hartigan. *Clustering Algorithms*. John Wiley and Sons, New York, 1975.

[Hea99] M. Hearst. Untangling text data mining. In *Proc. of ACL'99 the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[HHP$^+$01] S. Havre, E. Hetzler, K. Perrine, E. Jurrus, and N. Miller. Interactive visualization of multiple query result. In *Proc. of IEEE Symposium on Information Visualization 2001*, pages 105 –112. IEEE, 2001.

[Hid02] J. M. G. Hidalgo. Tutorial on text mining and internet content filtering. Tutorial Notes Online: `http://ecmlpkdd.cs.helsinki.fi/pdf/hidalgo.pdf`, 2002.

[HK97] M. A. Hearst and C. Karadi. Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchie. In *Proc. of the 20th Annual Int. ACM SIGIR Conference*, pages 246–255. ACM, 1997.

[HKLK96] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Newsgroup exploration with the websom method and browsing interface, technical report. Technical report, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1996.

[HKW94] M. Hemmje, C. Kunkel, and A. Willett. Lyberworld - a visualization user interface supporting fulltext retrieval. In *Proc. of ACM SIGIR 94*, pages 254–259. ACM, 1994.

[Hof01] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 41(1):177–196, 2001.

[Hon97] T. Honkela. *Self-Organizing Maps in Natural Language Processing*. PhD thesis, Helsinki Univ. of Technology, Neural Networks Research Center, Espoo, Finland, 1997.

[HSS03] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. In *Proc. IEEE Int. Conf. on Data Mining (ICDM 03)*, pages 541–544, 2003.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *European Conf. on Machine Learning (ECML)*, 1998.

[Kei02] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):100–107, 2002.

[KJ03] V. Kumar and M. Joshi. What is data mining? `http://www-users.cs.umn.edu/~mjoshi/hpdmtut/sld004.htm`, 2003.

[KKL$^+$00] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paattero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, May 2000.

[Kod99] Y. Kodratoff. Knowledge discovery in texts: A definition and applications. *Lecture Notes in Computer Science*, 1609:16–29, 1999.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Koh84] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[KOT$^+$04] J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity task at jnlpba. In N. Collier, P. Ruch, and A. Nazarenko, editors, *Proc. Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–76, 2004.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley, New York, 1990.

[KSB01] C.H.A. Koster, M. Seutter, and J. Beney. Classifying patent applications with winnow. In *Proceedings Benelearn*, Antwerpen, 2001.

[KZ02] M. Krier and F. Zacca. Automatic categorisation applications at the european patent office. *World Patent Information*, 24(3):187–196, September 2002.

[LAHF03] J.-C. Lamirel, S. Al Shehabi, M. Hoffmann, and C. Francois. Intelligent patent analysis through the use of a neural network: Experiment of multi-viewpoint analysis with the multisom model. In *ACL-2003 Workshop on Patent Corpus Processing*, 2003.

[LGMF04] Jure Leskovec, Marko Grobelnik, and Natasa Milic-Frayling. Learning sub-structures of document semantic graphs for document summarization. In *KDD 2004 Workshop on Link Analysis and Group Detection (LinkKDD)*, Seattle, Washington, 2004.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[LK02] E. Leopold and J. Kindermann. Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46:423 – 444, 2002.

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

[LMS91] X. Lin, G. Marchionini, and D. Soergel. A selforganizing semantic map for information retrieval. In *Proc. of the 14th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, New York, 1991. ACM Press.

[LS89] K. E. Lochbaum and L. A. Streeter. Combining and comparing the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing and Management*, 25(6):665–676, 1989.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[Mai02] R. Maitra. A statistical perspective on data mining. *J. Ind. Soc. Prob. Statist.*, 2002.

[MAP$^+$04] E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proc. 1st Conf. on Email and Anti-Spam (CEAS 2004)*, Mountain View, CA, USA, 2004.

[McC03] A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. Conf. on Uncertainty in Articifical Intelligence (UAI), 2003.*, 2003.

[Mer98] D. Merkl. Text classification with self-organizing maps: Some lessons learned. *Neurocomputing*, 21:61–77, 1998.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MM99] I. Mani and M. T. Maybury, editors. *Advances in Automatic Text Summarization*. MIT Press, 1999.

[MS01a] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 2001.

[MS01b] M. E. Mendes and Lionel Sacks. Dynamic knowledge representation for e-learning applications. In *Proc. of BISC International Workshop on Fuzzy Logic and the Internet (FLINT 2001)*, pages 176–181, Berkeley, USA, 2001. ERL, College of Engineering, University of California.

[NM02] U. Nahm and R. Mooney. Text mining with information extraction. In *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases*, 2002.

[NMTM00] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134, 2000.

[Nür01] A. Nürnberger. Interactive text retrieval supported by growing self-organizing maps. In Timo Ojala, editor, *Proc. of the International Workshop on Information Retrieval (IR 2001)*, pages 61–70, Oulu, Finland, Sep 2001. Infotech.

[Pd05] G. Paaß and H. deVries. Evaluating the performance of text mining systems on real-world press archives. In *Proc. 29th Annual Conference of the German Classification Society (GfKl 2005)*. Springer, 2005.

[PL02] P. Pantel and D. Lin. Document clustering with committees. In *Proc. of SIGIR'02, Tampere, Finland*, 2002.

[Por80] M. Porter. An algorithm for suffix stripping. *Program*, pages 130–137, 1980.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.

[RC01] D. G. Roussinov and H. Chen. Information navigation on the web by clustering and summarizing query results. *Information Processing & Management*, 37(6):789–816, 2001.

[RHM02] D.R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408, 2002.

[Rob77] S. E. Robertson. The probability ranking principle. *Journal of Documentation*, 33:294–304, 1977.

[Roc71] J. J. Rochio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System*, pages 313–323. Prentice Hall, Englewood Cliffs, NJ, 1971.

[SAB94] G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 37(2):97–108, Feb 1994.

[SB88] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[SEK03] M. Steinbach, L. Ertoz, and V. Kumar. Challenges of clustering high dimensional data. In L. T. Wille, editor, *New Vistas in Statistical Physics – Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer-Verlag, 2003.

[SJW97] K. Sparck-Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann, 1997.

[SKK00] M. Steinbach, Ge. Karypis, and V. Kumara. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000. (see also TR 00-034, University of Minnesota, MN).

[Sma99] H. Small. Visualizing science by citation mapping. *Journal of the American Society for Information Science*, 50(9):799–813, 1999.

[SP03] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. Human Language Technology NAACL*, 2003.

[Spo95] A. Spoerri. *InfoCrystal: A Visual Tool for Information Retrieval*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[SS99] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[SS00] R. E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. (see also TR74-218, Cornell University, NY, USA).

[TC02] K. Takeuchi and N. Collier. Use of support vector machines in extended named entity recognition. In *6th Conf. on Natural Language Learning (CoNLL-02)*, pages 119–125, 2002.

[TMS05] Text mining summit conference brochure. `http://www.textminingnews.com/`, 2005.

[UF01] A. Wierse U. Fayyad, G. Grinstein. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann, 2001.

[van86] C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986.

[Wil97] Y. Wilks. Information extraction as a core language technology. In M-T. Pazienza, editor, *Information Extraction*. Springer, Berlin, 1997.

[WMB99] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco, 1999.

[WTP$^+$95] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proc. of IEEE Symposium on Information Visualization '95*, pages 51–58. IEEE Computer Society Press, 1995.