Label Monitoring on Document Streams

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieurin oder Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: *Diplom Wirtschaftsinformatiker René Schult*
geb. am: *10.10.1975 in Lübz*

Gutachterinnen/Gutachter:
Prof. Dr. Myra Spiliopoulou
Prof. Dr. Yannis Theodoridis
Prof. Dr. Andreas Nürnberger

Magdeburg, den 4. Mai 2012

Contents

List of Figures

List of Tables

List of Abbrevations

CO_2	carbon dioxide
CFS	Concept Formation System
CFWS	Clustering based on Frequent Word Sequences
DHC	Dynamic Hierarchical Compact
DHS	Dynamic Hierarchical Star
EM	Expectation-Maximisation
FIHC	Frequent Itemset-based Hierarchical Clustering
HMM	Hidden Markov Model
KDD	Knowledge Discovery in Databases
KL	Kullback-Leibler
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
NLP	Natural Language Processing
NMF	Non-negative Matrix Factorisation
NMI	Normalised Mutual Information
OLDA	Online LDA
PLSA	Probabilistic Latent Semantic Analysis
SQL	Structured Query Language
TCT	Text Clustering Toolkit
TDT	Topic Tracking and Detection
TF$\times$**IDF**	Term Frequency $\times$ Inverse Document Frequency
UML	Unified Modelling Language
XML	Extendet Markup Language

1 Introduction

1.1 Motivation

The quantity of information continues accumulating about 1.5 billion gigabyte per year in numerous repositories [Durfee 08]. To analyse such big data the research area ”Data mining” has developed many different methods. Document repositories are a special form of such data repositories. A very large percentage of business and academic data is stored in textual format, like document collections or text archives. People grouping documents together on the basis of a priori unknown criteria. One of the most common and successful method of organising such huge amounts of documents is to hierarchically categorise documents according to topics or keywords [Kim 05]. Caused by the rapid growth of the document collections, manually analysis of the text data becomes nearly impossible. The usage of data mining, especially text mining methods were developed to fulfil the users information needs [Chung 05]. So automatic procedures are needed for grouping documents according to criteria which are not known a priori.

Parallel to data mining, which finds new patterns and trends in numerical data, text mining is the process to discover unknown patterns in free textual data [Kroeze 05]. Nasukawa and Nagano state that text mining ”is a text version of generalised data mining” [Nasukawa 01]. A special property of text data is that this data is not overtly structured data like standard (mainly numerical) data in relational databases, except the meta data like author, date, publisher and so on. Clustering texts, as a sub-part of text mining, consists of grouping text documents together in a cluster, which are very homogeneous in the group and the groups should be very heterogeneous. Especially in large text collections it is easier to assign labels to get an overview about the content of the clusters. Labels can also be helpful at the categorisation process of a document collection, because the documents of a cluster are similar to each other and therefore a label can represent a category point for example.

Those different categorisation problems in the document collections demonstrate different research problems on the collection categorisation itself and at the problem of the actuality of such a categorisation of document collections.

As the Internet grows, document collections become bigger and bigger [Dvorský 04, Kroeze 05]. They are no static collections, they evolve over time. The existing categorisation represent the collection of one time point. If over time, new documents are added to the document collection and a good category for the new documents does not exist, we have a problem at the adding process. One solution can be to add the documents to another existing category, instead of creating a new category. But now we have to update

the category itself, so that it represents correctly all documents in it. Another solution can be to create a new category for the new documents. With both possibilities we have to update the existing categorisation, so that they represent the correct content of the document collection.

In many applications the users are interested in seeing new trends evolving in the document archives. For example researchers are interested in new research areas which are showing up during the years or which ones gain fewer attention throughout the years.

In some application areas, reading such documents for the purpose of detecting new useful information is a very demanding task. For example, a so-called database user in bio-informatics must be familiar with the linguistic conventions and acronyms used and also has to possess up-to-date knowledge about the latest discoveries in this domain in order to be able to distinguish between the state-of-the-art and new contributions in any specific document. Similarly, a business analyst must be acquainted with the latest facts and activities in the market to be monitored and needs to be able to distinguish between already known facts and emerging trends. Another example for the growth of a text collection and the need to find changes in this collection is that research advances provide several examples to this end; topics like ambient intelligence or galaxy dynamics are rather young disciplines, while alchemy is not as popular as it used to be during the middle ages. As a collection of documents grows, the terminology may change as well. For example, made-up words like bubble sort and acronyms like Structured Query Language (SQL) and Extendet Markup Language (XML) have emerged in computer science, while words like PASCAL or COBOL have recently lost some of their earlier popularity. The effective acquisition of information from the whole of a growing archive requires the discovery and monitoring of topics that describe the archive contents at different points in time.

For example, business analysts are likely to be interested in the role that carbon dioxide (CO_2) emissions will play in strategic decisions in the years to come. It is reasonable to expect that concepts like *emission*, *environmental protection* and CO_2 will become increasingly important in relation to concepts like *logistics* and *transportation*. However, a central question remains: Which *terms* and *term combinations* are going to emerge and in which contexts? Even the simple concept CO_2 is associated with different terms in different contexts, and thus, searching in documents with an a-priori defined list of terms seems too restrictive for such an *emerging* subject. For the search term `CO2`, Wikipedia[1] returns an article on `carbon dioxide`.

The search term `carbon dioxide` itself returns several articles; the "list of countries by carbon dioxide emissions" is on position 11, the "list of countries by ratio of GDP to carbon dioxide emissions" is on position 16, followed by "photosynthesis" and then by the "list of countries by carbon dioxide emissions per capita" on position 18. The search term `carbon dioxyd` (sic!) returns a warning (because of the misspelled term) and three articles: the "Energy policy of China", "Energy in Japan" and the "Energy policy of the United States", in that order respectively.

[1] en.wikipedia.org, 27.01.2008, approx. 15:00 GMT+1

Both examples above show that it is not easy to handle with changing vocabulary in document collections over time. Problems are, how to deal with changes in the usage of the language or the importance of some terms or through new developments as the vocabulary used changes over time. The contents of document sources include text written in natural language, mostly in a domain-specific jargon. To deal with these problems, text clustering and label the clusters on the document streams can be one solution for it.

Much of this copiously extracted knowledge is stored in ontological resources, including collections of entities (genes, proteins, companies, patents, products, etc.), taxonomies of concepts and their relations (e.g. “X immediately activates Y”, “A and B are involved in a research alliance”, “F has submitted a patent on Z”). The document collections are ordered along side such a taxonomy. The value of such resources cannot be overestimated. However, the importance of specific contents may vary over time, as new subjects emerge and old ones may fade away in this categorisation.

To this purpose, we perform topic monitoring upon the document stream in this work, taking into account emerging and disappearing words that describe the topics [Schult 08]. For example topics in a text collection to one time point can be ”imaging”, ”mining” and ”knowledge discovery”. At a later time point the topics can be ”image mining” and ”knowledge mining”. This show, that the words describing a topic can change and also merged to an evaluation of a topic.

This leads to the following research questions about monitoring topics and their evolution over time in document archives as described in the next sub-chapter.

1.2 Research Questions

As discussed in the previous section, the categorisation of document collections consists of different problems. In the following we propose research problems for which a solution is presented in the next chapters.

- Building a method for creation and adaptation of clusters over an evolving feature space and label these clusters:

  A problem during the clustering of an actual document collection is the actuality of the clustering. Normally new documents are inserted in the document collection. All incoming documents can be ordered by the incoming time stamp and can be observed as a stream of documents. A clustering process over the document streams must be created under the condition that an evolving terminology exists, resulting in an evolving feature space over the document stream.

- Creating a method for label adaptation at clustering over an evolving feature space:

  Clustering labels of a document collection can be created as a kind of summary of the clusters. The problem here is the selection of the cluster label because such cluster label should be intuitive understandable and should be able to give an

overview about the content of the documents inside a cluster. Furthermore, such cluster label should also be persistent over a number of periods. One option for selecting a cluster label is to create common topics over the documents in a cluster or to select the frequently used words within the documents of a cluster. The usefulness of the cluster used at the document collection decreases and should be updated, equally a self designed cluster or a taxonomy can be used for the grouping of the document collection. The interesting research question at this step is to select the right point in time for updating of the labels and with which word the used label should be adapted.

- Finding a method for tracing clusters and their labels along the time axis:

  The interesting research question at this point is how can we trace clusters and their labels over time before and after an update step. At some points in time the existing clusters and labels can be updated (see problems described before). But also conditions exist which cause a new clustering and/or labelling step at the given time point. At this point problems exist, in tracing the clusters and labels from the previous point in time to the actual point in time.

- Defining an evaluation methodology for cluster and label monitoring:

  The research area about cluster and label monitoring over a document stream is a new area, no evaluation standard exists for it. An evaluation has to be defined, a method to evaluate the quality of the monitoring process.

In this thesis we developed a solution to solve the problems with cluster and label adaptation, tracing labels over time and evaluating the quality of this label monitoring process over document collections. In the following sub-chapter we briefly introduced our solution.

1.3 Research Methodology

How can labels of document clusters in a document stream be monitored to find out changes in topics of the documents, arising topics and also vanishing topics? This interesting research question can be used to summarise the research problem of this work. This overall research objective is henceforth addressed by (i) establishing a conceptual framework for cluster adaptation and monitoring labels of document clusters over a document stream, the ”ThemeFinder”, (ii) developing a research prototype that implements the monitoring algorithm and an user interface, to create a proof-of-concept also supporting the evaluation step and (iii) evaluating the quality of the monitoring process by a real-world text archive.

To answer these research questions, different research areas are addressed like adaptive clustering, stream clustering, incremental clustering, cluster labelling methods, label adaptation, label monitoring, frameworks for cluster comparison, evolving feature spaces

and updating feature spaces too. This shows the inter-disciplinarity of our research and the necessity to combine those areas to create a solution answering these research questions.

In the conceptual framework, a document collection over time is modelled as a document stream, where each document has an incoming time stamp. This document stream is divided into time slices and the documents of each time slice are clustered. A new paradigm for document stream clustering is developed, considering (1) accumulating data and (2) sliding windows, as alternative strategies for data forgetting.

For the application of clustering algorithms, the documents has to be transformed into a vector representation so that the clustering algorithm can use the data. The feature space can change over time because new documents can be added to the document collection or the taxonomy of the documents changes over time. If the vector representation should be a good representation of the content of the documents, the feature space must adapted to the new features. Otherwise the changes in the documents cannot be recognised. For example if a new topic arises in new documents, but the words of the new topic are not represented in the feature space, the clustering and labelling process cannot detect this new topic at the document collection. That is why we have included a solution for the adaptation of the feature space over time in our research solution.

So a set of cluster labels for each time slice is created. Now the set of labels for each time slice can be compared and similar labels or detect changes of labels between different time slices can be found. This process is called a monitoring process at cluster labels. This fits in the problem with tracing labels. During the adaptation of the feature space, which has an influence to the adaptation of the labels too, which fits in the problem about adaptation of labels. As one result of the monitoring process of the cluster labels new emerging labels, vanished labels but also a change of labels; especially the changes of a label reflect the different evolution of the authors language can detected.

Labels or sub-labels, which are present at a number of time slices can be candidates for adaptation of the existing categories, create new categories or adapt the taxonomy used. Here the domain expert has to define how many time slices a label or sub-label should be present so that it can be a candidate for such category or taxonomy adaptation.

For this label monitoring process we have developed the ”ThemeFinder” to handle the cluster labels at different time slices and monitoring the cluster labels especially to recognise the changes of the cluster labels at the different points in time.

The developed process which takes as input an archive of documents described by a (small) set of terms and associated to a single topic of the taxonomy. We cluster these documents on term similarity and derive topics that can serve as cluster labels. Then, as new documents are added, we re-consider the clusters and the feature space of terms, upon which the clusters are built. Topics persisting over several periods of time, several re-clusterings and feature spaces are good candidates for the taxonomy. Groups of words associated to a given topic during a given period are good candidate keywords for searching on this topic in this period, independently on whether the topic is later added to the taxonomy or not.

Now we give a short overview about the structure of this work.

1.4 Outline

Chapter 2 (Relevant Topics) gives an overview of clustering and text mining. We decide to use clustering the solution of this work and give a short introduction to clustering algorithms, which we have used during the experiments, following the knowledge discovery basics in chapter 2. Different methods exist to create a label for a document cluster. After the clustering part an overview about such labelling methods is given. After this basics about knowledge discovery and the methods which we have used in our framework, we give an overview about other methods at the tangent research areas like evolving topics in text clusters and monitoring of cluster evolutions at non-textual data. This gives an overview about relevant research in this area.

Afterwards **Chapter 3 (A Framework for Monitoring Cluster Labels over a Document Stream)** defines the fundamental terminology and definitions necessary to understand the ”ThemeFinder”. After this we present a detailed view of our monitoring framework for cluster labels in document streams, the ”ThemeFinder”. This includes the solution to handle changes at the taxonomy and the vocabulary at the document collection by methods for updating the feature space and doing the adaptation of the clusters and the labels over the time. The monitoring framework includes also a method for comparing cluster labels as part of the label monitoring process over the given time periods.

Ultimately, the evaluation of our framework can only be done with experiments with real-world datasets, especially document archives. Therefore, **Chapter 4 (Experiments)** covers different experiments to show the evaluation process of the whole framework. Here we describe the experiments with different forgetting strategies including experiments to compare the framework results to different clustering algorithms or different labelling methods. This chapter includes also a comparison to the FOCUS framework, which is a well known framework for cluster comparison.

As dataset for the experiments, we use a public available document archive, a subarchive of the ACM archive.

For the experimentally evaluation of the framework ”ThemeFinder” specified in chapter 3, a functional prototype system needs to be implemented, that supports all features of ”ThemeFinder”. **Chapter 5 (TheMoT - the Theme Monitoring Tool)** gives an overview of the prototype system developed as an integrated part of this research. Chapter 5 outlines the system requirements and describes the architecture of the prototype developed, the Theme Monitoring Tool. Subsequently, this chapter includes a description of the core functionalities of Theme Monitoring Tool. This tool supports not only the evaluation of our framework, it supports also the usage of this framework, because it is developed from the perspective of the user of this framework.

Chapter 6 (Conclusion and Outlook) summarises the contribution of this work and

indicates promising research challenges to enhance the framework and improve the capabilities of ”ThemeFinder” and Theme Monitoring Tool.

2 Relevant topics

2.1 Basic Literature

In this chapter we will present shortly the basic literature, to the main domain of the topic of this work, like data mining, text mining clustering or labelling methods. After this we give a more detailed overview of the relevant research to monitoring changes at streams and specially document streams.

2.1.1 Data Mining

Data mining as term is often used as synonym for knowledge discovery in databases [Roiger 03]. As Fayyad et al. define in [Fayyad 96] ”knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”. A similar definition can be found in Giudici [Giudici 03] but they use the term data mining. This shows very clear that both terms are very often used as synonyms. This research area is a very interdisciplinary topic at the interface of statistic, machine learning and database systems [Ester 00]. It can also be seen as Knowledge Discovery in Databases (KDD) process, illustrated in figure 2.1.

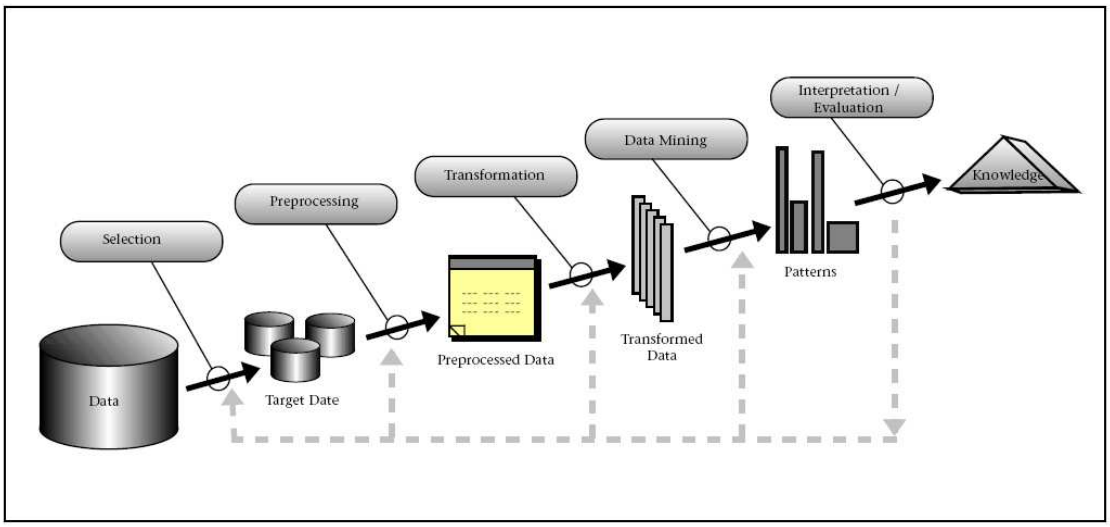


Figure 2.1: The Data Mining Process [Fayyad 96]

The KDD process consists of the steps selection, preprocessing, transformation, data mining and interpretation/evaluation. At this process the topic data mining alone is the

usage of the specific methods to extract the knowledge from the data as one part of the whole KDD process. It can be seen that this process is an iterative process and results from one step can also influence decisions at previous steps so that the process go further at this previous step. For a detailed description at each step I refer to the cited references.

At the data mining step the actual data mining or data analyse will be done with the specific data mining methods. This methods are groups for instance in association rules, classifications and clustering [Petersohn 05]. In this thesis clustering is used at the datasets and explained later in chapter 2.1.3 the clustering and some algorithms.

A special application part of data mining is the new research area text mining.

2.1.2 Text Mining

Text mining is the process of finding unknown patterns from free texts. Parallel to the data mining process, which finds patterns and trends in numerical data [Kroeze 05]. Most of the knowledge at business is normally stored at text collections and to find the needed information from this collections text analysis is very useful [Durfee 08]. Text mining is more than only searching through meta data and full-text databases to find existing informations. Nasukawa & Nagano say that text mining should ”focus on finding valuable patterns and rules in text that indicate trends and significant features about specific topics” [Nasukawa 01]. Text mining can be used for instance at academic research for scanning large numbers of literature to order the most relevant documents or to extract the important topics from a huge document collection to order the documents.

One basic difference from the text mining process to the standard data mining or KDD process are the tasks at the data preparation phase. Caused by the data which should be analysed the free texts need to be transformed to a computer understandable representation. These preprocessing operations centre on the identification and extraction of representative features for natural language documents [Feldman 07]. These data preparation steps are for instance the removing of punctuations, or remove words which are not very informative like stop words or expletive words [Mladenic 05]. Another step is the so called stemming which is the process of reducing the morphological variants of the words to their stem or root like transform plural forms of a word to their singular form [Moens 00]. The last step after cleaning up the text from not so useful words and punctuations is to transform the texts to one representation that enable applications of the desired text-mining methods. The word-vector representation, introduced by Salton in [Salton 89] as vector space model, is one of the simplest and most frequently used representation of texts. It is also called the bag-of-word representation. The idea behind is simple, for each word in the text, ignoring their ordering and any other textual structure, the word-vector contains a weight proportional to the number of its occurrences in the text [Mladenic 05]. One example of such a weight is the term frequency or combined with the inverse document frequency. To create these word-vectors as representation of the text it is needed a feature space which consist of all words which are important for the collection and define so the length of the word-vectors. A simple possibility to create such feature space is to use all words of the document collection. One other possibility is to define a list of important words to the domain of the text collection by a domain expert.

This has the advantage of a much shorter feature space, which reduce automatically the size of dimensions and so save time at the usage with the text mining algorithms. The disadvantage is the need of an domain expert which means extra costs to select the right words for the feature space. A mixture of both approaches is the usage of the n dominant words of an automatic generated feature space, so the dominant words are used but the size of dimensions is not so high compared to the first approach.

2.1.3 Clustering

The learning from data comes in two flavours the supervised learning and the unsupervised learning [Gentle 04]. Unsupervised learning is closer to the exploratory spirit of data mining. One unsupervised group of mining methods is clustering. The goal of a clustering is that the objects within a group be similar to one another and different from the objects in other groups. If the homogeneity in a group and the heterogeneity to other groups is greater, the clustering is better [Tan 06, Giudici 03, Ester 00, Kruschwitz 05]. Clustering is a relative old research at the data mining research. So it exists different types of clustering algorithms. One typecast of the different clustering algorithms can be the partition into hierarchical cluster algorithms, partitioning algorithms and density based algorithms.

At the hierarchical algorithms a cluster can have sub-clusters and so a set of nested clusters can be build as an organised sub-tree [Tan 06]. The hierarchical clustering algorithms build a cluster hierarchy known as dendogram [Huang 05]. The partitioning algorithms divide the data set into non overlapping subsets so that each data point is exactly in only one subset. The most known algorithm of this type is the k-means clustering algorithm [Hand 01]. Later I will explain this algorithm shortly. The third type of cluster algorithms is the group of density based algorithms. The basic idea of this algorithm is that the data points are group at regions with a high density and other regions with a lower density of the data points. The local point density at each data point at a cluster is not over a threshold. The local point density of a point is defined as the number of data points in a defined region around the data object [Ester 00]. The well known DBScan algorithm from Ester et al. in [Ester 96] is one example of this cluster type.

At the following I will shortly introduce the cluster algorithm k-means and bisecting k-means as an example of the partitioning algorithms and the DBScan algorithm as an example of the density based algorithm because later at the experiments we have used both algorithms.

2.1.3.1 K-Means and bisecting K-Means

The most used partitioning clustering algorithm is the k-means algorithm because it is a very simple and easy to use algorithm. The core idea of the k-means is that each cluster is assigned by a centroid and each data point is assigned to the nearest centroid.

The pseudo code of this algorithm is shown in table 2.1 [Tan 06]. At Step 1 the algorithm select K data points as initial centroids for the K clusters. Than each data point is

assigned to the nearest centroid (step 3). After this the centroid of each cluster is recompute in step 4. Step 3 and 4 now will repeat till the centroids do not change or another stopping criteria like maximum iterations is reached.

Step	Action
1	Select K points as initial centroids
2	**repeat**
3	From K clusters by assigning each point to its closest centroid
4	Recompute the centroid of each cluster
5	**until** Centroids do not change

Table 2.1: basic k-means algorithm

The k-means algorithm has different problems. One is the problem how to set the initial value for K and which data points should be selected as starting points for the initial centroids. A selection of poor starting centroids for example could lead to a non global optima of clusters of the data [Berkhin 02].

The bisecting k-means algorithm is a straightforward extension of the k-means algorithm. It is based on the idea to split the data set into two clusters and then select one cluster and split again, and so on, till the selected number of K is reached. So it produce a hierarchical tree over the dataset and the cut of the tree will be done at the selected number of clusters.

In table 2.2 [Tan 06, Karypis 00b] the algorithm is shown.

Step	Action
1	Select a cluster to split
2	find 2 sub-clusters using the basic K-Means algorithm (bisecting step)
3	**repeat** step 2 (bisecting step) for ITER times and take the split which produce the clustering with the highest overall similarity
4	**repeat** step 1-3 **until** the selected numbers K of clusters is reached

Table 2.2: bisecting k-means algorithm

To find the cluster to split it exists different ways. One possibility is to use the largest cluster or the cluster with the least overall similarity [Karypis 00b].

At the experiments described in Chapter 4 we used the bisecting k-means as one cluster algorithm.

2.1.3.2 DBScan

In contrast to the partitioning cluster algorithm, now the density based DBScan cluster algorithm from Ester et al. [Ester 96] will be introduced in a more detailed view. This algorithm is based on the number of points in a defined region around one data point.

The minimum number and the radius of the region has to be defined by the user. This are the $minPts$ value and the eps region value. Informally described the algorithm checks for each point if it is a core point or not. A core point is a point which have minimum $minPts$ data points in his eps region. Two core points that are close enough, means the distance is maximum the eps distance, are put into the same cluster. All points which are no core points but they are in an eps region of a core point are put to the same cluster as their core point and are called border points. Points which are not in any eps region of a core point are called noise points [Ester 96, Tan 06].

A formal detail of the algorithm is given at table 2.3 [Tan 06].

Step	Action
1	Label all points as core, border or noise points
2	Eliminate noise points
3	Put an edge between all core points that are within eps of each other
4	Make each group of connected core points into a separate cluster
5	Assign each border point to one of the clusters if its associated core points

Table 2.3: DBScan cluster algorithm

The advantage of this cluster algorithm is that it is a very simple and effective density based algorithm. It can find clusters with large density and also clusters with lower density. Also this algorithm has no problems to handle noise points, like the k-means algorithm.

2.2 Monitoring Changes at Text Streams

In the book about data streams from Aggarwal [Aggarwal 07], the author introduced the problems of handling data streams and the different types of algorithms on data streams, like clustering, classification or frequent patterns. Also Aggarwal gives an overview of data preparation steps and forgetting strategies to handle data streams. The more recent book of Joao Gama [Gama 10] has similar intension to the book of Aggarwal including a detailed description of some algorithms.

2.2.1 Topic Tracking and Detection – TDT

The subjects of Topic Detection and Topic Tracking are defined in [Allan 02], where the five tasks of TDT are enlisted. As stated in that book, TDT concentrates on the detection and tracking of stories (a ”topic” is a story) and encompasses the tasks of (1) story segmentation, (2) first story detection, (3) cluster detection, (4) tracking and (5) story link detection. There is a conceptual similarity between TDT and the identification of emerging topics in a (noisy) document stream, in the sense that the emerging classes to be

discovered are ”topics”. However, these classes are not stories in the TDT sense. It is not of interest to detect a story and then track it across documents, as in tasks (2) and (4), but rather to identify documents across different time periods, which, when taken together, contribute to the same, a priori unknown but statistically important ”topic”. This separation has been elaborated first in the survey of Kontostathis et al. ([Kontostathis 03]), where the new task of topic trend discovery was introduced. However, we have explained in [Schult 06b], the methods presented under this task in [Kontostathis 03], including Pottenger & Yang [Pottenger 01] and Roy, Gevry & Pottenger [Roy 02], rely on cross-references among documents, i.e. on task (5) of the original TDT agenda, and thus do not transfer to a stream of independent documents that do not cite each other.

Instead of the usage of typical classification methods like in the TDT project, Anaya-Sanchez et al. use clustering methods to detect topics at a document collection in [Anaya-Sanchez 10].

Anaya-Sanchez et al. developed a different document clustering method to detect and describe topics of document clusters. They make the assumption that each topic can be created by pairs of words from the document collection and that there is no knowledge about the document collection. Which is very typical for document streams for example. For a given document collection $C = \{d_1, d_2, ..., d_n\}$ their algorithm creates a set of clusters $G' = \{(\delta_1, G_1), ...(\delta_m, G_m)\}$ where a cluster $G_i \subseteq C$ represents a topic with topic description δ_i. Starting with the most probable word pair π they calculate a set $C|\pi$. If this set is homogeneous to the content of the cluster, the authors create a cluster with the relevant documents $G = Rel(\pi)$ for the given π and create also the description $\delta(\pi)$. If a cluster is created, its documents are deleted from C. If the cluster is not homogeneous, the word pair is rejected. This procedure is repeated till the document collection is empty or no relevant word pairs exist. For each document of the document collection, which is not a member of a cluster, a single set with the most frequent word pair is created. This method is a static method to find topics at a document collection. The created clusters are not overlapping. The authors show at evaluation that their methods produce better results as the Frequent Itemset-based Hierarchical Clustering (FIHC) [Fung 03] or the Clustering based on Frequent Word Sequences (CFWS) [Li 08].

The TDT methods and the methods from Anaya-Sanchez are useful on static datasets, but in this work we concentrate on dynamic datasets here.

In contrast to classification and clustering at a document collection, Kontostathis et al. present [Kontostathis 04] a new emerging trend detection algorithm which is based on Latent Semantic Indexing (LSI). They used LSI for dimension reduction. They perform the singular value decomposition process of the LSI, the term by document matrix into three matrices: T, a term-by-dimension matrix, S a singular-value-matrix (dimension by dimension) and D, a document-by-dimension matrix. The original term-by-document matrix can be obtained through matrix multiplication of T, S and D. The LSI method is adapted in the way that only the k largest singular values at T and D are used. The goal of this step is to reduce the noise inside the data. After this step, the authors compute the term-term similarity. They consider two possible ways to calculate it, create the term-term matrix using TS or use the term-dimension matrix and apply a vector similarity like the cosine similarity. The authors preferred the cosine similarity because it had a better

performance at different experiments. The cosine similarity compute results in the range $[-1, 1]$. Kontostathis et al. developed a sparcification algorithm to reduce the values of this similarity results. They delete all values which absolute value is near zero, because this values give relative small informations. So they reduce nearly 70% of the values without any degradation in retrieval effectiveness. After this step the authors start with the term clustering algorithm. A cluster is created for each term i in the collection: terms similar to it are put into a cluster. The complete process needs only two input parameters, k for the LSI process and a threshold which determines the size of the clusters. The authors show that their solution produces better recall values and comparative at precision values as the sLog algorithm, also an emerging trend detection algorithms with cluster creation.

Another topic detection method on text documents was developed by Whithney et al in [Whitney 09]. In contrast to the methods before, they do not concentrate on detecting all topics at the document collection, but on finding surprising events at the topic lists i.e. they detect only topics that are different from those, detected before. The approach of Whitney et al. [Whitney 09] analyses a text stream to detect surprising events. They define three different types of surprising events, (1) the point discontinuity, where the event appears only once, (2) the jump discontinuity, where a surprising change in all text documents arises (3) as third the slope discontinuity, where the change is slow. To detect this three types of events in a text stream, Whitney et al. classify the documents of the text stream and develop different surprise statistics, based on the χ^2-test to detect unexpected events based on the words of the classifier. One algorithm of the surprise statistics is the Pearson-method. Given is x_t as the number of documents which contains the word in time point t and N_t as the number of documents at the same time point. The authors compare the relative number of x_t in N_t with the number of x_t in all previous time windows. Another method is likelihood ratio: the likelihood ratio for a hypothesis is the ratio of a maximum value of the likelihood function over the subspace represented by the hypothesis, to the maximum value of the likelihood function over the entire parameter space. Another option is the Gaussian statistics: it compares the observed value x_t with the average of the previous values x_{-t}, normalized by the standard deviation of the previous values in $x_{-t}(s)$. As fourth and fifth method Whitney et al. combine the Gaussian statistics with one of the χ^2 methods. At the experiments they show that the Gaussian statistics and their combinations with the χ^2 test have very similar results. Both other methods (the Pearson and the likelihood ratio) return results of lower quality.

All methods are able to detect topics at a document collection. But their process is very different, apart from the basic methods classification, clustering and LSI, because at some of Whitheys methods you need values from the reality to compare with the results of the algorithms and adjust it. The second point of differences is the different definition of a topic or that is directly the goal to find out as topics. For our research goal all this methods are not helpful, caused by different reasons, only for static datasets or different definitions of a topic or not the monitoring goal behind the detection.

The next chapter takes a more detailed look to different ways to defining a topic. Mentioned before it has an influence to the results at topic detection, how to define the topics. As examples a topic can be the dominant words of a group of the most used

words, or the most describing words of a document or a group of documents. Ultimately, a topic is a non-empty label over a set of documents, e.q. a cluster label. At the next chapter we discuss about some methods for cluster label creation.

2.2.2 Cluster labels

If a text collection is processed as data set and clustering algorithms are used, clusters of documents will be the result. A fundamental goal of text clustering is the identification of topics present in the text corpus. A second objective is to stimulate human interpretation of the clustering results [Greene 07]. This leads to the identification of the clusters' meaning. A widespread practice is to present labels that reflect the semantic content of each cluster [Sanfilippo 04].

The easiest and most used method for label creation is the top-ranked method. It produces a term weight for each document and term. The h terms with the highest values at the term weights are used as terms for the label of each cluster [Greene 07]. These labels provide a summary of the cluster content. A problem is that the same terms may be selected as labels for multiple clusters. Further the set of chosen terms may be too generic in nature, not very specific for a topic.

Hence, Greene [Greene 07] proposes two further labelling methods that interpret the task of selecting discriminative terms for each cluster as a feature selection task.

1) χ^2 measure (CHI): The Chi-square test as labelling method is used to identify terms that occur frequently in one cluster but rarely in other clusters. Based on the standard $\chi 2$ formula the matrix $W \in \mathbb{R}^{m \times k}$ is calculated with following equation:

$$W_{ij} = \frac{n * (a_{ij}d_{ij} - c_{ij}b_{ij})^2}{(a_{ij} + c_{ij}) * (b_{ij} + d_{ij}) * (a_{ij} + b_{ij}) * (c_{ij} + d_{ij})} \tag{2.1}$$

where t_i is the i-th term and C_j is a cluster. a_{ij} is the number of documents in cluster C_j that contains the term t_i. b_{ij} is the number of documents in other clusters that contain t_i. c_{ij} is the number of documents in cluster C_j that do not contain t_i and d_{ij} is the number of documents in other clusters that do not contain t_i. Similar to the information gain method, the labels are generated by choosing the highest values in each column of the matrix W.

2) The second exploits information gain, as proposed by Yang & Pederson in [Yang 97]. Given the term membership matrix U, measure the number of bits of information needed to distinguish between the cluster based on a low or high value in U for a given term. The following equation calculates the weight for the i-th term in the cluster C_j:

$$W_{ij} = E_{ij} - \frac{1}{k}\sum_{l=1}^{k} E_{il} \tag{2.2}$$

where k is the number of terms, and the entropy E_{ij} is given by

$$E_{ij} = -U_{ij}\, log_2\, U_{ij} - (1 - U_{ij})\, log_2(1 - U_{ij}) \tag{2.3}$$

The labelling for a cluster C_j can be found by selecting the h terms with the highest weights in the j-th column of W.

The usage of these methods depends heavily on the application, which topics should be created and which definition for a topic and cluster is preferred at the application.

In Neill et al. [Neill 05] and Yang et al. [Yang 05], clusters are geometric objects that move or change shape in a metric space. Neill et al. study the emergence and stability of clusters, observing spatial regions across the time axis [Neill 05]. However, their notion of ”cluster” is very particular: A cluster is a region where counts (for some property) are higher than expected. This notion cannot be used for topic evolution, because a document cluster is rather a region of objects that are more similar to each other than to the rest of their neighbourhood (cf. cluster definition against a baseline [Mei 05]).

The described methods before are useful for clusterings there all clusters are on the same level, which means non-hierarchical clusterings. For hierarchical clusterings it exists some special methods to create the labels for each cluster because hierarchical clusters have special requirements to the labelling process, such as that the label must be distinguish a cluster from the sibling clusters and it must be show the difference between the cluster and its parent cluster.

Popescul and Ungar present an automatic labelling method which should be not so complex like typical distribution functions. In [Popescul 00] they introduce a χ^2-method which based on a significance test for stochastic independently. They build a cluster hierarchy of the document collection and than they put all words of a cluster into a container, so the connection between words and documents is lost. After typical data preparation steps on document collections, like stemming or stop word removal, they start with a significance test for each word in a node against all child nodes in the hierarchy starting at the root node. If the hypotheses, that a word is present in all child nodes, is correct, the word will be put into the label container of the cluster and it will removed from all cluster containers of the child nodes. If all words are checked and the label container of a cluster contains more words, a defined number of words with the best test results will be selected as label for the cluster. This process will be done for each node of the cluster hierarchy and as result they get a hierarchy of cluster labels as a representation of the document hierarchy. Popescul and Ungar present a second method, the frequent and predictive words method, for labelling a document cluster which is based on the well know Term Frequency $\times$ Inverse Document Frequency (TF$\times$IDF) value. As input also a clustering of the documents is needed, but it must not be a hierarchical one, like the method before. They calculate the local frequency of a word for a cluster multiplied with the TF$\times$IDF value with this formula: $p(word|class) \times \frac{p(word|class)}{p(word)}$. The $\frac{p(word|class)}{p(word)}$ is similar to the TF$\times$IDF value and the $p(word|class)$ is the local frequency of a word for a cluster. The authors test both algorithms at an example data set from research papers of the cora search engine. The created labelled are evaluated only by three computer science students.

Two other methods for label a hierarchical clustering are presented by Bade et al. in [Bade 07]. The first method is a labelling method with knowledge of the real classes for the clusters. The method starts at the root node of a hierarchy and label each cluster. The authors set a minimum precision value for a class in all labelled items and if a label is found which follow the defined precision criteria, the label is set to the node of the hierarchy. The second method use not known class labels. The labels will be created based on a term statistics, called descriptiveness, which use a document frequency for each term and each cluster for the parent and also the child node too.

2.2.3 Frameworks to handling streams

The shown methods for topic detection are only for static data sets. But today the main document collections are not static and the topics at such a collection can chance over time. That's why in this work we concentrate on handling a document collection as a data stream. Here first we give an overview about existing methods to handle data streams. A data stream is a sequence of data points $\{x_1, x_2, ...x_n\}$, which can be read only in this order. The reading process is called "linear scan" and only a limited number of scans are possible. To respect the different researches at this area and present a better overview about this research area, we divide the methods into two parts, to handle data streams and document streams. To get knowledge from such data streams the algorithms can be divided into to parts, the adaption methods, which adapt an existing model to represent the actual data. On the other hand we have the monitoring frameworks, which monitor changes at the models over time by handle the data streams. Both sections are divided again into methods which are for data streams and methods which use the specific of document streams.

2.2.3.1 Adaption of a model

In general an adaption of a model is used at a stream of data, it exists always a model over old data and the model will be adapted with new incoming data, so that the arising model is a good representation of the actual data.

Data Streams Generic methods for cluster evolution have been published under the labels "incremental clustering" and, more recently, "spatiotemporal clustering". The latter methods usually assume the existence of a stationary trajectory with an associated metric. An early work on the detection of cluster change and the adjustment of spatial clusters has been proposed in Ester et al. [Ester 98]. Ester et al. used the term "incremental clustering" for a method that re-computes the clusters after each update of the dataset, paying emphasis on the minimization of the computation overhead. They proposed IncrementalDBSCAN, an adaptive variant of the static DBSCAN proposed earlier by the same group in [Ester 96]. We have already described the static DBSCAN algorithm in this chapter. IncrementalDBSCAN focuses on cluster adjustment. Nonetheless, the authors propose a typification of cluster changes [Ester 98]: When a new object is inserted, this may cause the creation of a new cluster (a formerly small neighbourhood

becomes adequately large), cluster absorption (an existing cluster absorbs the new object and its neighbours, if any) and cluster merger (the members of different clusters become density-connected). When an object is deleted, a cluster may shrink or even disappear (some of its objects lose the core property, so that the neighbourhoods connecting its members disappear) or become a split (the fragments of the original cluster are disconnected but adequately large to become clusters themselves). Since noise is a primary characteristic of the documents we consider, methods that are robust against noise are of particular interest. However, DBSCAN has been designed for application areas where proximity of data records is independent of the temporal dimension, i.e. the distance between two data records cannot change from one time point to the next. This holds for spatial databases, including geographical information systems (GIS). Distance among documents might be defined in a similar way, e.g. using Euclidean distance or (more usually) cosine distance. However, the feature space across a document stream is not constant, since some terms become obsolete while others emerge. Hence, IncrementalDBSCAN is not trivially applicable.

A very elaborate method for cluster evolution has been proposed by Aggarwal in [Aggarwal 05]. In his approach, a cluster is a densification of the topological space and is described by a kernel function. The emphasis of [Aggarwal 05] is on studying the velocity of change in an evolving cluster and on identifying (a) the dimensions of the feature space, which are most responsible for change and (b) areas or data points that exhibit the highest velocity of change. In the context of topic evolution, this method can be used to identify areas that evolve at different speeds and associate sets of words (labels) with them. However, the constraint of a static, a priori known feature space applies for this method similar to IncrementalDBSCAN. The method of Aggarwal is from the core point different to the IncrementalDBScan because Aggarwal will detect topics at the document collections with a different chance speed as the other topics. This means topics which new arise or topics with a decreasing importance. This topic definition and monitoring goal are also different to our definitions of a topic and our monitoring goal.

Guha et al. developed the STREAM algorithm, which read a defined number of data points m of the data stream and cluster this data points. The clusters of the clustering result will be weighted based on the content of each cluster. To reduce the memory space, they store only the medians of the cluster and their weights. This will be done till $m^2/2k$ objects are read. The medians are clustered into $2k$ clusters. Based on the number of m, the available memory will be used optimal. For the clustering step of the STREAM algorithm, Guha et al. developed a new algorithm, the LSEARCH. The basic idea behind is the variable value for the number of clusters, the value k, instead of the K-means algorithm, there k is fix. The LSEARCH create a new cluster or not based on a probability of the distance to the nearest cluster and the saved costs for each data point. After this new creation the points around are new assigned. The main target of this is the costs at the given solution. Is the number of clusters between k and $2k$, an end solution is found. [Guha 03]. The advantage of this method is the possibility to handle relatively fast a data stream and all data points are used for the model creation, no forgetting. But during the clustering of the medians and the resulting medians cluster again, the deep of possible informations from the clustering is reduced.

Zhong presents an online version of the spherical k-means algorithm in [Zhong 05a]. The spherical k-means is based on the cosine similarity to cluster high dimensional text documents. The online version of the spherical k-means, the OSKM, is based on the "winner-takes-all competitive learning", which incrementally updates the cluster centroid at every adding of a new document. The goal is to increase the cluster quality of the spherical k-means but have the same efficiency at the data set. The used WTA strategy updates only one cluster centroid after insert a new document. Adding a new data point x, incrementally the cluster centroid $\mu_{k(x)}$ will be updated by $\mu_{k(x)}^{(new)} = \frac{\mu_{k(x)}+\eta x}{||\mu_{k(x)}+\eta x||}$ with η as learning rate and μ as normalization to the vector space. Zhong show that his method is more efficient as the self-organizing maps or the neural gas methods. A problem of this method is, that only the cluster centroids are stored and so no access to the cluster members is possible, which is needed, if we will label the resulting clusters after the clustering step. An other problem can be, that the distance between two clusters becomes very small so that it normally can be better to merge both clusters. Zhongs method do not recognize this problem.

Another solution to actualize existing clusters over a data stream is developed by Tasoulis and Vrahatis in [Tasoulis 05]. Their method is based on the k-windows algorithm and extends this method to handle dynamic databases without an overhead at calculation time. The adapted k-windows method is used to find the clusters. They try to group all data points of a cluster into a d-dimensional window. The methods movement for centering the window over the data points and the enlargement method will be used at this step. The enlargement method try to increase the size of the window to maximize the number of data points in the window. Depending on the overlapping part of different windows, some windows can be merged together. The found clusters will be stored at a Bkd-trees. This allows update processes like insert and delete, which are needed at operating with a data stream. The advantage of this method is the small calculation time and the performance of the algorithm. A disadvantage is the high complexity of the algorithm itself. The understanding of the windows creation and the merging of windows in some cases is not very easy to understand and not very intuitive.

The IncrementalDBSCAN algorithm is very intuitively and can handle also merges and splits of existing clusters after insertion or deletion of data points. The algorithm of Aggarwal is more useful to detect special changes, which are different to the direction of the other parts of the model.

Document Streams Document streams are special data streams, because the data objects at the stream are documents, which need a special preprocessing as normal data, like stop-word removal, NLP processing, stemming and so on. The second point at document streams is the changing vocabulary, there for example new words can arise or the meaning of words can change too. This are some reasons that's why the contemplation to the special data type documents at streams is done now.

Gil-Garcia and Pons-Porrata present in [Gil-Garca 10] two hierarchical algorithms for clustering document streams. Both algorithms are dynamic algorithms. The Dynamic Hierarchical Compact (DHC) and the Dynamic Hierarchical Star (DHS) algorithm based

on the dynamic hierarchical agglomerative and the updating of the max-S graph algorithm. Both used the β-similarity and the maximum β-similarity to create graphs of the document collection. The graphs are created like a hierarchical cluster algorithm from bottom up by cluster the existing graphs of the previous step. The nodes at the graphs represent a cluster. The dynamic hierarchical agglomerative algorithm actualize both graphs, the β- and the maximum β-similarity graph. For each added node the similarity to each other node is calculated. An edge will be created if the β-similarity is over the threshold β (β-similarity) or if the β-similarity is the maximum value (maximum β-similarity). The updating of the S-max graph updates the max-S graph independently from the insert or delete commands. Based on this, the DHC algorithm allow the separation of two connected graphs. The DHS has the addition feature that overlapping of clusters is allowed. The time complexity of the DHC (DHS) is $O(n^2 * m)(O(n^3 * m))$ and the needed space is $O(nm)(On^2 * m))$. At experiments the authors compare the F-measure and an adapted precision measure of both algorithms with the UPGMA and the bisecting k-means algorithm. The DHS has better F-measure values caused by the overlapping and the relative high number of clusters. The DHC produce significant better results at the adapted precision value and is more efficient as the DHS.

Another solution to handle concept drift via hierarchical clustering is presented by Widyantoro and Yen in [Widyantoro 05]. They developed a framework, FEILDS, which extends an existing concept drift algorithm to learn drift also from sparsely labelled data. They use a set of relevant unlabelled data to compensate the small number of labelled data. To identify the relevant unlabelled data they need knowledge about the concept of each instance of the labelled data stream, the Stream-L. FEILDS use a concept hierarchy because the concept extraction is very difficult from a small set of data points in Stream-L. They use the Concept Formation System (CFS) to create the concept hierarchy which cluster the input stream into a hierarchy with their concepts. The Concept Drift Tracker will used by the concept drift learner only if it is needed. The concept drift tracker use the Stream-L and the concept hierarchy to create a new Stream-S there all data points are ordered by the income time extended with the concepts. A normal concept drift learner can be used now. The authors show that the FEILDS framework increase the performance of existing algorithms. A disadvantage is the extra calculation costs.

Both hierarchical methods use hierarchical clustering algorithms to cluster the documents of a stream and adopt this hierarchical structure during the handling process of new documents. The problem of finding topics at hierarchical structures is to find clear topics for a cluster that is different to the topic of the parent and the child nodes but should also represent the hierarchical structure. Both methods are useful to detect topics over time via adaption their existing hierarchy, but both algorithms need extra calculation time and the topic creation is not very easy.

Instead of it, Aggarwal and Yu [Aggarwal 06] rather derive content summaries for clusters over accumulating streams. They introduce the notion of ”droplet” as a statistical summary of data that is stored and inspected at regular intervals [Aggarwal 06]. In particular, a droplet consists of two vectors, one accommodating co-occurring pairs of words and one accommodating the words occurring in the cluster and their weights. The

members of a cluster are weighted and contribute to the clusters weight. This weight is part of the droplet and subject to a decaying function: A clusters weight decays if no new points are added to it. Aggarwal and Yu [Aggarwal 06] achieve the identification of new clusters by juxtaposing cluster droplets and new data: A new document is assigned to the cluster, whose droplet has the highest similarity to it, according to some similarity/distance functions. A cluster becomes ”inactive”, if no documents are added to it for some time. If a document cannot be assigned to a cluster, then it becomes a new cluster itself replaces the oldest inactive cluster. Hence, new data form new clusters while old clusters decay gradually if they are not fed with new documents. This approach conforms to the intuition behind topic detection and tracking in the conventional TDT sense: A new topic is a document that does not fit to any existing topic; a topic decays and disappears if no new documents on it arrive any more.

In the method of Aggarwal and Yu [Aggarwal 06], cluster summary is not associated with semantics: A droplet is a condensed cluster representation appropriate for matching and maintenance but not necessarily intended for human inspection and interpretation. Methods on the monitoring of cluster labels are rather assuming that a label is a human-understandable representation of a cluster content; accordingly, they focus on the evolution of the semantics captured in the cluster label.

The clustering of text streams is also considered in [Zhong 05b], albeit the emphasis is on adapting the clusters rather than detecting changes in their labels. Shi Zhong proposes an online variation of K-means, the ”online spherical k-means”: documents are modelled as TF$\times$IDF vectors, normalized into unit length, whereupon the clustering algorithm builds k clusters, maximizing the average cosine similarity within each cluster [Zhong 05b]. A new document is assigned to the cluster with the closest centroid, whereupon the cluster itself is adjusted. Cluster labels are not derived nor studied by the algorithm itself and are only used for the evaluation of the algorithm upon pre-labelled experimental data. The advantages and disadvantages of this method is similar to the standard OSKM algorithm, discussed before already. Zhong say nothing about the definition of topics and the question, how to adopt the topics of a cluster after each step or after a defined time step, is not defined to actualize the topics of the clustering.

Different to this, the next authors concentrate more on topic detection and evolution. In the topic evolution mechanism of Moringa and Yamanichi, a topic consists of the words with the largest information gain [Moringa 04], in contrast to the droplets from Aggarwal. The topics reflect the contents of soft clusters, built with an incremental Expectation-Maximisation (EM) algorithm. Finite mixture models are learnt at each time stamp and dynamic model selection is performed to choose the optimal one. The key idea of this model selection is first to built a large number of components and then select the main ones on the basis of Rissanens predictive stochastic complexity. The emphasis of their work is on the adaptation of the topics rather than the tracing of topic changes.

The tracing and interpretation of topic changes is studied by Mei and Zhai [Mei 05]. Similarly to Moringa & Yamanichi [Moringa 04], they consider mixture models to build document clusters and also use the Expectation-Maximisation algorithm to this purpose. Hence, a document may belong to multiple clusters and, consequently, topics describing

different clusters may overlap. To derive the topics themselves (as combinations of words describing a cluster), they assume a background model. Then, the evolution of topics is traced using Kullback-Leibler divergence, similarly to Ipeirotis et al. [Ipeirotis 05]. Mei and Zhai introduce topic transition types and build a topic evolution graph, in which transitions are traced using Hidden-Markov-Models [Mei 05]. A remarkable aspect of the topic evolution graph is that edges/transitions are not restricted to topics derived at consecutive periods: A topic is connected to any topic discovered at an earlier period, if the former turns to be a transition of the latter (according to KL-divergence).

The usefulness of both methods described before depends on the definition of topics and the used application, if the Expectation-Maximization step is useful or not.

Instead of cluster the document stream and create topics from the clusterings, Blei and Lafferty use a complete other method to analyse the topic evolution over time. They present in [Blei 06] a method for analyse the time evolution of words and topics in document collections. Based on their own statistical content model, the Latent Dirichlet Allocation (LDA) algorithm, they replace the dirichlet allocation with the gaussian distribution to handle dynamic data. After creating time slices over the document stream and using the variant of the LDA, the topics must be defined at the first time slice. Each topic will be represented by a vector with the natural parameters of the topic. This allows the usage of a multi-nominal distribution. The authors assume that the topics flow the gaussian distribution over time. This calculated values are used in a simplex-algorithm, an extension of the normal distribution. Blei and Lafferty introduce an α into this distribution to model the uncertainty. Now they combine the topics with their distributions. The result is many static topic models which are timely connected to some others. After this, the authors work with variational methods. The idea behind variational methods is to optimize the free parameters of a distribution over the latent variables in the way that the distribution is close to a Kullback-Leibler (KL) divergence to the true posterior. Then this distribution can be used as a substitute for the true posterior. They used the variational Kalman Filtering and the Wavelet Regression at this step. At the experiments Blei and Lafferty show that both methods have advantages depending on the special target. One problem of this method is the fix number of topics at the beginning and that the topics must be known, so this method is not useful for our problem.

Topic evolution can also be studied with methods that discover latent models instead of clusters, namely with Probabilistic Latent Semantic Analysis (PLSA) [Hofmann 01] and Latent Dirichlet Allocation [Blei 03]. A related approach is [Mei 05], where PLSA is independently applied to document batches of different time points. The temporal local PLSA models are linearly combined with a simple static unigram model, which models background words and is empirically estimated on the overall set of all documents seen in the stream so far. They use KL-divergence to compare topic-word distributions found at different not necessarily adjacent time points and connect similar topics into an evolution graph. On this graph, life cycles of themes (as topic sub-graphs) are analysed with a Hidden Markov Model (HMM). This method uses a fixed static vocabulary, i.e. assumes that all words are known in advance, and also considers a single background model over the whole time horizon.

AlSumait et al. [AlSumait 08] propose Online LDA (OLDA). They extend the Gibbs

sampling approach suggested by Griffiths and Steyvers [Griffiths 04] to handle streams of documents. Gibbs sampling at one time point is used to derive the hyper-parameters of the topic-word associations at the next time point, so that successive LDA models are coupled. New words are collected as they are seen, so AlSumait et al. do not assume that the whole vocabulary is known in advance. A topic is represented as a vector of probabilities over the space of words. The dissimilarity between topics can be computed using the Kullback-Leibler divergence. For two data points p_1 and p_2 the authors use the average of $KL(p_1||p_2)$ and $KL(p_2||p_1)$ because the KL is not symmetric. They see an emerging topic as the one that is different from its peer in the same stream or from all the topics so far. AlSumait et al. developed the algorithm "Edetect" to detect the emerging topics, there the methods before are included. The emerging topic detection via "Edetect" is one step of the complete Online LDA algorithm. By processing small subsets of documents only, the OLDA algorithm is able to learn meaningful topics with similar quality or in some cases better as LDA. The main problem of the OLDA algorithm is the different input parameters, like the confidence level, the weight vector and the Dirichlet values.

LDA has also been used to find scientific topics [Griffiths 04] together with temporal properties, like cold and hot topics. Incremental LDA [Song 05] updates the parameters of the LDA model incrementally as new documents arrive. However, the algorithm also assumes a fixed vocabulary and additionally it does not forget the influence of past documents and old outdated words. The incremental LDA was evaluated among other algorithms in [Banerjee 07], however it was outperformed by the online von-Mises Fisher mixture model, which is a generalization of spherical k-means. The dynamic topic model [Blei 06] partitions the time axis and uses a basic LDA model for each partition. The hyper-parameters of the LDA models are propagated over time via a state space model similar to a HMM. However, the approach also assumes a stationary vocabulary and does a backwards analysis over long periods. A non-markov approach to study topics over time is proposed in [Wang 06], which extends the LDA model to generate the time stamps of documents also. This causes the founded latent topics to concentrate on time periods, where the vocabulary used in documents is homogeneous. Topics end, when the used vocabulary shifts, so the approach can not model the evolution of the vocabulary within topics.

The methods above with LDA show that topic detection and evolution is also possible with these methods. They are useful for this task, but for the end users it is not so easy to understand the probabilistic models and understand intuitive the topics, which will be produced as result of these methods. Also some methods have the restriction of a static vocabulary or have no forgetting methods included.

2.2.3.2 Monitoring Frameworks

For finding changes at topics or monitor the changes at the data over time the adaptive models are not very useful. The adaptive models have every time an actual model but not really recognize the changes from the previous model to the actual one. Monitoring frameworks compare the previous and the actual model to detect the changes in it, so that

the user is able to find out the reasons for the changes.

Data Streams Between 1999 and 2000, Ganti et al. proposed three modules for the observation and analysis of changes in datasets [Ganti 99b, Ganti 99a, Ganti 00]. They observed three components of one framework, these modules can be used to detect and monitor evolution in datasets or clusters over them, as well to derive and monitor summary descriptions over the data. Ganti et al. [Ganti 00] proposed DEMON for data evolution and monitoring across the temporal dimension. DEMON detects systematic vs. non-systematic changes in the data and identifies the data blocks (along the time dimension) which have to be processed by the miner in order to extract new patterns. In the context of topic evolution, DEMON delivers a mechanism for the selection of those documents that should be considered for the discovery of new clusters. Hence, DEMON can be observed as a mechanism that specifies the data to be forgotten and those to be remembered at each point of time.

The module FOCUS [Ganti 99b] proposed by the same group compares two datasets and computes an interpretable qualifiable deviation measure between them. The deviation is represented in the form of models consisting of a structure component and a measure component. The structure component identifies interesting regions and the measure component summarizes the subset of the data that is mapped to each region. Clustered datasets are a special case: Clusters are non-overlapping regions, where each region is described through a set of attributes (structure component) and corresponds to a set of raw data (measure component). This elaborate and powerful mechanism can split the clusters under comparison down to identical regions and thus provide an overview of their differences.

The ”Pattern Monitor” (PAM) [Baron 03] models patterns as temporal, evolving objects. A model of changes is more recently proposed in Baron & Spiliopoulou [Spiliopoulou 04]. The main emphasis of PAM is on the monitoring of association rules with a more recent extension for clusters. However, topic monitoring is beyond its scope.

Another recently published framework is MONIC for the monitoring of cluster evolution [Spiliopoulou 06]: MONIC encompasses a model for cluster transitions, such as a cluster being split or absorbed by another or changing in size or homogeneity. Its notion of ”overlap” among clusters captured at different time points allows for changes in the feature space, thus becoming appropriate for the task of topic evolution over a stream of documents. Indeed, MONIC has been tested on an evolving document collection, the ACM Digital Library section H2.8 (this section contains also the publications on data mining as subsection). Although MONIC can by nature be used to detect emerging topics, it has not be designed for interaction with the human expert: It lacks a visualization method that intuitively captures topic evolution and assists the human expert in following the traces of topic splits and merges. Especially for the study of noisy document collections, such an assistance seems to be indispensable, so it is incorporated in our approach presented later.

The frameworks DEMON, FOCUS, PAM and MONIC concentrate to one specific type of data mining models, clusterings or association rules. All together need an over-

lapping window of partitions of the data stream, to find corresponding data points in both models from the overlapping windows. To monitor the parts of mining model, they take a look to the data itself, so they need a storage of all the data of the monitoring phases.

The PANDA framework is a more generalization to compare complex and simple patterns. This framework [Bartolini 04] delivers mechanisms for the comparison of simple patterns and aggregation logic's for the comparison of complex ones. In the PANDA framework, a simple pattern is built upon raw data, e.g. a clustering or a set of association rules, while a complex pattern consists of other patterns, e.g. a cluster of association rules. Hence, the comparison of complex patterns and the subsequent computation of the dissimilarity score between them is performed in a bottom-up fashion. A complex pattern is decomposed in component patterns which are compared to each other. Then, the dissimilarity scores are combined according to a user-defined aggregation logic. In terms of expressiveness, PANDA subsumes FOCUS, as explained in [Bartolini 04]. The PANDA framework is based on other frameworks which must be used to compare the simplex patterns.

Different to the frameworks before with comparing different clusterings via overlapping data objects, it exists a method, which is graph-based by Yang et al. They detect change events upon clusters of scientific data [Yang 05]. They study "Spatial Object Association Patterns" (SOAPs), which are graphs of different types, e.g. cliques or stars. A SOAP is characterized by the number of snapshots in the data, where it occurs and the number of instances in a snapshot that adhere to it. With this information, the algorithm detects formation and dissipation events, as well as cluster continuation. The types of cluster evolutions are also relevant for topic evolution, but the methodology itself does not transfer, because it requires the establishment of links among the objects under observation.

The problem of concept drift is a specific problem at handling data streams. The frameworks described before monitor all changes and are not specialized to concept drift. Dries and Rückert notice, that concept drift recognition can be seen as a statistical hypothesis test with two tests of multivariate data. If the distribution of the last data is different from the data before than we can talk from concept drift. In [Dries 09] the authors present three new concept drift detection methods. The first method is the CNF Density Estimation Test and based on the density estimation of a binary presentation of the data. They transform the normal data into many binary vectors for each feature. The comparison of the vector set for all given data points with the vector set of the data points at a different time point can give an significance value for the difference between both data sets and so they can give a value for the concept drift inside the data. The second method is the SVM-Margin-Test. This method is based on a PAC_Bayesian-Analysis of a linear classifier, which is induced by the first data set and evaluated by the second data set. The main goal of this method is to find out a weight vector of the attributes between both data sets like the 1-norm SVM. This method is also called the margin-method. The third method based also on the SVM, but used the error rate instead of the distance. Dries and Rückert use two different error rates, the 0-1-loss and the sigmod-loss-function. Both functions will be used as statistic test to evaluate the SVM results in [Dries 09].

Document Streams The specials of document streams at data streams are also be covered by monitoring frameworks, specially to monitoring topics and their evolution over the time.

Leskovec et al. will find text segments and will detect their evolution at a news web site [Leskovec 09]. Their presented method is different to many others. They used as document collection the complete collection over the time span and cluster the phrases of the documents. Their idea is to produce clusters at this kind, that a cluster contains a text phrase and its variations at the complete time span. The clustering algorithm works as follows. All text phrases are compared to each other, to find out relations or inclusion relations with very small mismatches of phrases. They build an acyclic directed graph by create connections from the short to the longer phrases. All edges get weights from the frequency of the phrases. Now all phrases are ordered and can be divided into clusters. The edges with the smaller weights will be deleted. This deletion process will be done in the way that at the end each cluster has only one root node. In this case a root node is a node which has only incoming but no outgoing edges. To monitor the phrases over time the authors analyse a cluster over time, because all phrases have its time stamp and so each cluster can be tracked over the time line and analyse the size for example. The investigation of the size of each cluster gives the authors a feeling about the actual important topics at the document collection. The authors say nothing about the time and resource complexity of their presented method.

Leskovec's method is based on phrases of the documents, so for a document it can consists of many phrases and becomes part of many different clusters after the clustering process. The advantage is, that documents can have different topics in it and this topics can be found by this method. But the structure of the documents is broken and a monitoring of the topics from one document is not possible. Also the complete data stream will be used as one dataset at the analyzation step, so the content of the complete stream must be stored.

Ipeirotis et al. trace content summary changes, whereas a "content summary" refers to a database of documents [Ipeirotis 05]. The motivation is that content summaries are valuable for the selection of the databases that should be queried in the first place. Despite the difference in focus between their work and the issues addressed here, the methodology is transferable, since a content summary, as defined in Ipeirotis et al. [Ipeirotis 05] has similarities to a cluster label. The authors distinguish between a *complete content summary C(D)* that consists of the number of occurrences of each word $w \in D$, i.e. $f(w, D)$, and an *approximate content summary* $C(D)$ computed by document sampling. To predict, when a content summary change occurs, Ipeirotis et al. [Ipeirotis 05] apply survival analysis, a method originally designed to model/predict the length of survival of patients under different treatments. In particular, they define the "survival time of a summary" as the time until the current database summary is sufficiently different from the old one. To predict this survival time, they consider (a) a measure of difference between summaries at time points and (b) a probability distribution that is expected to be followed by the change. The measures or "content summary change indicators" they use to this purpose are the Kullback-Leibler divergence, i.e. the difference in the word distribution between the old and the new summary, and the sim-

pler ”precision” (defined as the number of words in the current summary that are also in the old one) and ”recall” (defined as the number of words in the old summary that are also in the new one). The probability, with which a content summary changes, may occur within time t is assumed to follow an exponential distribution $S(T) = e^{-\lambda t}$, in which the value of λ should be predicted for a specific database. In their experiments with text collections from the Internet domains GOV, EDU and COM, Ipeirotis et al. assessed differences in the speed of change. The GOV collection changes more slowly than the others, while COM and other commercial site collections change faster than the rest. Moreover, large databases tend to change faster than small ones. Ipeirotis et al. [Ipeirotis 05] consider content summaries over whole databases of documents, the topics of one document itself are not so interested. So they can monitor the main topics of the whole collection but not sub parts inside the collection or special topics covered only by a small group of documents .

Ling et al. present in [Ling 08] a framework to detect topics into text collections. They create a two step model with a bootstrapping method to extend the aspect keywords and a probabilistic model at this to calculate the word distribution for each aspect. They define d as document of a collection C with $c = (w, d)$ with w as word in document d. The facet model θ in the text collection C is a multi-nominal distribution of words $\{p(w|\theta)\}$ which represent the aspect model. The multi-faceted overview of a topic is a semi-structured summary of all informations about a queried topic, which is structured in the way that sentences are grouped into the most relevant facets. The last definition is the multi-facet overview mining. Given are the definitions before and user defined keywords to give a semi-structured overview of the query topics and present it with the user-specified facets. At first Ling et al. initialize the aspects by construct an undirected graph of terms where each node is a term and each edge indicates the similarity relation between the both terms. To initialize the aspects the nearest neighbours at the graph are calculated. The authors propose two methods to model the aspects, the PLSA model and a log-likelihood function which they maximize with an EM algorithm. The EM algorithm is used to estimate topic models. They use previously initialized facet models to define a prior on the facets and estimate the models using the maximum a posterior (MAP) estimator. Based on this the authors generate an overview. The documents which are relevant for the queried topic, are divided into sentences. Now the relevant values by the models are calculated between the different sentence pairs and only the highest values represent an aspect. At the end each aspect is represented by a rank list of the included sentences.

Another view of document streams and the definition of the topics which should be monitored is presented by Mei and Zhai. They show in [Mei 05] a method based on evolutionary graphs to monitor themes over time. A theme in their view is topic in a text collection with a probabilistic distribution of words from an unigram language model. Their main goal is to extract a theme evolution graph automatically from a document stream. First they divide the stream into possible overlapping sub-collections. Than the most salient themes from each sub-collection are extracted using a probabilistic mixture model. As third step for each theme in two sub-collections decide whether there is an evolutionary transition based on the similarity. To extract themes, Mei and Zhai use a

probabilistic mixture model there words are regarded as data drawn from the mixture model with component models for the theme word distribution and a background word distribution. They have introduced theme unigram language models for each theme and for the whole collection a background model. To find out the evolutionary transitions, the authors use the Kullback-Leibler-divergence to calculate the evolution distance between two theme spans. They assume that two theme spans have a small evolution distance if their unigram language models are closer to each other. Based on the resulting theme evolution graph the authors can calculate the theme life cycles for the themes, which is defined for each theme as the strength distribution of the theme over the entire time line. At this step the authors use a Hidden-Markov-model with $k + 1$ states, k for the number of extracted themes and the plus one for the background model. The unknown parameter set in the Hidden-Markov-model can be estimated using an EM algorithm called Baum-Weich-algorithm. Then Mei and Zhai use the Viterbi algorithm to decode the text stream to obtain the most likely state sequence. As last step they use the sliding window mechanism to measure the strength of each theme at a time point. The authors evaluate their method with two real dataset. On both cases their methods can generate meaningful temporal themes. The usage of this framework depends on the definition of topics, if at the application a topic is represented by a distribution model or not. Normally in applications the topics should be user readable and interpretable.

Moringa and Yamanishi discuss a topic analysis framework in [Moringa 04]. With this framework the authors try to solve the three main tasks in Topic Tracking and Detection, (1) topic structure identification, (2) topic emergence detection and (3) topic characterization. Documents are represented by their TF$\times$IDF vectors in this case and they suppose that a text document has only one topic. For the topic identification task, Moringa and Yamanishi use a variant of an incremental EM algorithm to learn the finite mixture model at the documents from a specific time stamp. The topic structure in the text stream must be learned in an online fashion. Topic emerging detection is conducted by tracking the changes of main components in the mixture model. Topic characterization is conducted by classifying each text into the component for which the posterior is largest and then by extracting feature terms characterizing the classified fields. To find the topic emergence, the authors create a defined number of finite mixture models and select the optimal model for this time stamp with the dynamic model selection. This optimal model can be compared with the optimal one of a previous time stamp. They can recognize if the number of topics change from one model to the next, like new topics are created or some topics are lost. The last step, the topic characterization will be done by calculating the information gain of possible words and select the words with the highest value as characteristic words for the topic. Moringa and Yamanishi define topics as a mixture model from the EM algorithm. Such a mixture model changes over time and they detect the changes with their framework. The monitoring process is based on the mixture model which is created on the data objects itself.

All models and frameworks before are based on the monitoring of the data objects itself. The framework of this work is different to these methods and will be presented at the following chapter..

Summary In this chapter we introduced to the data mining area, specially to clustering and labelling methods. After this, we concentrated us to the area of stream mining, presented different methods from the side of adaptive algorithms and from the monitoring part. We divided both parts into the algorithms for data streams and specially for document streams. Different advantages and disadvantages of the algorithms to solve our problem of this work are shown.

3 A Framework for Monitoring Cluster Labels over a Document Stream

Our here presented algorithm ”ThemeFinder” undertakes the tasks of discovering clusters over a document stream, assigning labels to them and then monitoring the labels as the clusters change over time. First we present the basic idea of our approach, then we introduce the formalism and the definitions constituting to our model and then go ahead to describe the components of our framework, ending with the description of our ”ThemeFinder” algorithm itself.

3.1 Core Idea

The basic idea of our framework ”ThemeFinder” is to monitor cluster labels of a document collection over time. Monitoring of labels in our approach is to observe the labels over this document stream to detect changes at the labels from one time period to the next. So the framework consists of two parts. The first part is the clustering and labelling process at a document collection over a time period. During this process we will create groups of similar documents at the document collection and will create labels as a kind of summary about these clusters. This process results to a set of labels for each time period. A time period in our observation is a relevant partition of the document collection at the time axis from the application domain which should be done during observations. If a label changes over the time, this change will be raised by changes at the content of the cluster. Thus it is possible to reveal changes on the content of the cluster through monitoring the existing label of the cluster for distinct moments on the stream.

The second part of our framework is the monitoring process. This process takes as input two different sets of labels from clusterings at different time periods and monitors the labels from the one time period to the label set of the second one.

We describe now the labelling process and the monitoring process.

Labelling Process To cluster text documents of a document collection first we must transform the documents into vectors (represented by node vectorisation in figure 3.1). Before this vectorisation can be done, standard natural language processing methods like stemming and stop-word removing must be done at the documents (see node Natural Language Processing (NLP) precessing in figure 3.1) and a feature space, which is

needed for the vectorisation, must be created (represented by node Feature Space Creation in figure 3.1).

The complete labelling process to cluster a document collection and create labels is illustrated in figure 3.1 as an Unified Modelling Language (UML) activity diagram.

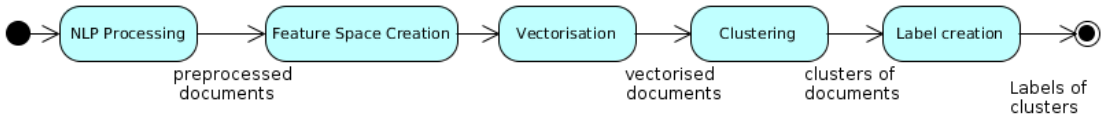


Figure 3.1: labelling process

At the end of this process the output is a set of labels for the clustering for this time period. The basic idea of the algorithm is to find the labels from the first clustering again at the set of labels of the second clustering. Cluster labels deliver a concise overview of the cluster content and so the set of cluster labels of a clustering gives an overview about the content of the clustered document collection. In contrast to this, clustering the documents and taking a detailed look at each cluster content is another possibility to get an overview about contents of a document collection. But this method is very time consuming and not useful in practice to get a fast overview about such content.

Monitoring Process The monitoring process detects changes in the labels from one time period to the next; this implies matching the labels found at different time periods. The complete monitoring process is illustrated in figure 3.2 also as an UML activity diagram.

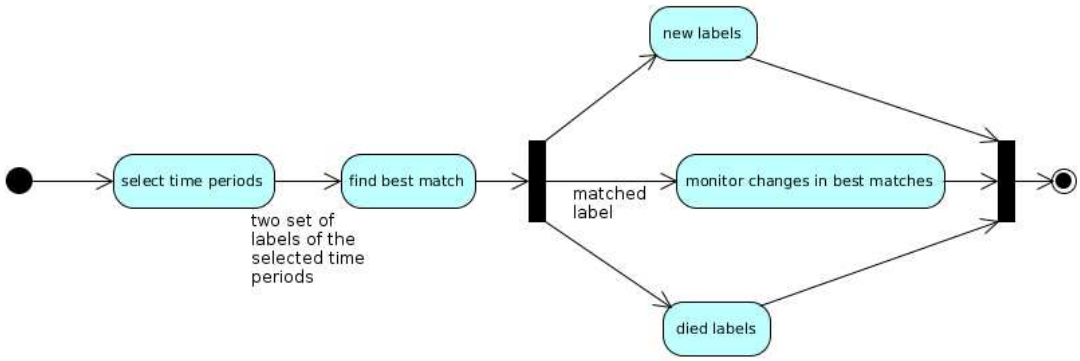


Figure 3.2: monitoring process

This monitoring process starts with the selection of the two time periods, which should be monitored by the framework. As result of this selection it exists two sets of labels of the selected time periods. The next step is the method *find best match*, which identifies, for each label of the label set from the first selected time period, the best corresponding

label of the label set of the second selected time period. As result of this matching the problem is reduced to three cases. The old label may be found again or not, in the latter case, it is called that the label is died. Of course, there may also be new labels that have no match among the old ones; they correspond to the new topics. We describe the detailed matching method later in chapter 3.3.2.

So recovered labels can be traced over time and they can be monitored to recognise changes at the labels over different time periods.

The monitoring process can detect labels or sub-labels that are persistent over many periods. These can be perceived as subjects that are of importance for a long term view and can be candidates for adaption or extension of the existing taxonomy for the document collection.

After having given the core idea, in the following we provide the definitions which are necessary for understanding the developed algorithm "ThemeFinder" .

3.2 Definitions

Let $\mathcal{A}$ denote a document archive, which will be observed over time. Further, let $\mathcal{W}$ be the set of all words in $\mathcal{A}$.

The archive $\mathcal{A}$ is observed over a series of T time periods $t_1, \ldots, t_T$. It exists two options regarding how many documents should be remembered and processed at each time period. On the one hand it exists the possibility to use the dataset of the actual period including all documents of the period before plus all documents which are new in the actual period. Let: $D_T = \mathcal{A}$ and $D_i \subset D_j$ with $i < j$. D_{in} is the set of documents which are new in the period t_i to $\mathcal{A}$. The index n is for the new documents so it is $D_{i+1} = D_i + D_{(i+1)n}$. We call this dataset the accumulated dataset.

On the other hand it exists the non-accumulate case. Here each period t_i encompasses a subset of documents D_i, such that $\cup_{i=1}^{n} D_i = \mathcal{A}$ and $D_i \cap D_j = \emptyset, \forall i \neq j$. Hence, in each period t_i, the document set D_i contains the documents that have been inserted in the archive during this period. This case is comparable with the typical forgetting method of sliding windows, in this case with the special property, that the windows size has the value one.

To create a feature space manual and automated methods can be used. For the manual method the knowledge of domain experts is needed, because the domain expert has a detailed knowledge about the researched domain and knows which terms are interesting or important at this domain. The creation itself is a very time consuming and cost expensive process. An alternative is the automatic creation.

For the automatic creation of the feature space, the easiest way is to use all words of the documents as a feature, but this leads to extreme large document vectors. To use only the *top-n* words of a document collection as a feature space is a possibility to handle this problem, which we use also in our framework. Remember that the feature space creation is a step after the data preparation phase and so all stopwords for example are already removed from the document collection.

At our framework we used the automatic feature space creation with the following definition of a feature space.

Definition 1 (Feature Space) *Let D be a collection of documents and W be the set of words in them. The "feature space" over D, $FS(D)$ is the set of the n "dominant" words in W, which is defined as the words with the highest TF×IDF values in D. So it is $FS(D) = \{w_1, ..., w_m\}$. For each time period $t_i, i = 1 \ldots, T$ the "period-specific feature space" $FS_i \equiv FS(D_i)$ is defined as the set of dominant words over D_i.*

By this definition, the size of the feature space remains constant across the time spectrum of observations, although the features of the period-specific feature space may change from period to period. Intuitively, labels should be derived at each t_i on the periodic-specific FS_i but this would by imply rebuilding the feature space, so "ThemeFinder" avoids this unless necessary.

The feature space is used to create document vectors.

Definition 2 (Document Vector) *Let D be a set of documents and $fs = \{w_1, \ldots, w_n\}$ be a feature space. Then, for each document $d \in D$, its "document vector" in fs consists of the TF×IDF values of the words in fs over D:*

$$v(d, fs) =< tfidf(w_1), \ldots, tfidf(w_n) >$$

By this definition, a document can be associated with several vectors, one per feature space. In particular, for each document d in the document set D_i of period t_i, we can define the document vector of d over the period-specific feature space FS_i or over another feature space $FS_j, j \neq i$.

The document vectors of the document collection D_i of a time period t_i will be clustered with a clustering algorithm to a clustering ζ which is defined as follows:

Definition 3 (Clustering) *Let D be a collection of documents and fs be a feature space. A clustering ζ is defined as a set of clusters that partition D into k groups of similar document vectors over the feature space fs.*

$$\zeta(D, fs) = \{C_1, \ldots, C_k\} \text{ with } \forall C_i, C_j \in \zeta : C_i \cap C_{j \neq i} = \emptyset$$

A labelling mechanism will be used to derive labels for the clusters in the actual clustering. The intention of a label is to provide concise description of the content of a cluster.

Definition 4 (Labelled Cluster) *Let D be a document collection, fs is the feature space and ζ the actual clustering like the definitions above. A cluster $C \in \zeta(D, fs)$ is a labelled cluster, if the following set is not empty:*

$$L_C = \{w \in fs | label(w, C) \geq \tau_{wordsupport}\} \neq \emptyset$$

L_C is defined as the label of the cluster C. A part of the set of words in L_C is defined as a sub-label for the cluster. The function $label(w, C)$ calculates for every word within cluster C a value that reflects the importance of this word for the cluster. If this value is greater as the user defined threshold $\tau_{wordsupport}$, then the word w is a part of the label L_C of the cluster C. One example for the function $label(w, C$ could be the fraction of documents in cluster C that contains the word w, divided by the number of all documents in cluster C, $card(C)$. We use this function with this interpretation for label creation at our experiments later.

This definition is similar to the labelling method "top-ranking", with creates a label as the n most used words in a document cluster. Labelling methods exist, which are based on the lose relation between the selection of discriminative terms as cluster labels on a fixed set of clusters and the task of feature selection in classification task. One is the "information gain" labelling method and one is the "χ^2 method". We have explained all three methods already in the literature overview in chapter 2.2.2.

A label should have different features like uniqueness and summary of the content [Stein 04]. The notions of label and thematic cluster reflect the insights [Karypis 00a] of concept indexing and of of latent semantic indexing [Deerwester 90]. Both studies agree that the importance of a component can be derived from the weights it receives in the analysis. Here the components are words, which we rank on their support within each cluster. The most frequent words inside a cluster constitute the cluster label.

A label may appear in only one period. In the literature the name *topic* is often used as synonym for a label. Now we introduce the term "persistent theme" first and "theme" thereafter for labels which appear more often than once. A persistent theme is defined as follows:

Definition 5 (Persistent Theme) *Let $t_1, \ldots, t_T$ be the series of T periods of observation over the document archive $\mathcal{A}$. Let D_i denote the set of documents in period t_i, FS_i be the feature space used in this period and $\zeta(D_i, FS_i)$ be the clustering of D_i over FS_i.*

*A set of words $TP \subseteq \mathcal{W}$, chosen among the words of the archive $\mathcal{A}$ is a "*persistent theme *" if:*

$$\forall i = 1, \ldots, T \exists C^i \in \zeta(D_i, FS_i) : label(C^i) = TP$$

A label that persists over all periods is a topic that characterises the data in a long term, so it is worth adding to the taxonomy or ontology of the application. However, the conditions defining a persistent theme are rather restrictive. So we define a less restrictive term, the theme according to Def. 6 after the following observations. First, we expect that a set of words would make a good theme if it appears in an adequately large number of periods m. Second, the terminology associated with a theme is not static: Especially during a peak of activity on a new theme, terminology may change rapidly as authors are looking for representative terms *and* as the borders between the new theme and other subject areas are being redefined. For example, the subject area now known as "data mining" used a slightly different terminology (and dominant terms) in 1995 compared to now. This indicates that the label of clusters that refer to the same theme may undergo changes. Therefore, we relax some of the requirements of Def. 5 as follows:

Definition 6 (Theme) *Let $t_1, \ldots, t_T$ be the series of T periods of observation over the document archive $\mathcal{A}$. Let D_i denote the set of documents in period t_i, FS_i be the feature space used in this period and $\zeta(D_i, FS_i)$ be the clustering of D_i over FS_i.*

As before, let $TP \subseteq \mathcal{W}$ be a set of words chosen among the words of the archive $\mathcal{A}$ and let $m \leq T$ be a threshold value. The set TP is a "theme" if there are m periods $t_{i_1}, \ldots, t_{i_m}$ such that:

$\forall j = i_1, \ldots, i_m \exists C^j \in \zeta(D_j, FS_j) : card(TP - label(C^j) \cap TP) \leq \tau_{deviation}$

This definition specifies that a label is a theme if some of its words appear in at least m arbitrary, not necessarily consecutive periods. The threshold $\tau_{deviation}$ determines how many of the words may deviate. By setting $\tau_{deviation} := 0$ and $m := T$, it results to the original definition of a persistent theme in Def. 5.

Different implementations and variations from this heuristic-based definition are possible: For example, it may be required that a minimum number among the m periods are consecutive or that m refers to the *last* m periods $t_{T-m}, t_{T-m+1}, \ldots, t_T$. It may be also required that a cluster C_j may be come a TP only if the most frequent word in L_{C_j} is in TP, thus restricting the candidate clusters considered for each TP at each period. The function $extract\_themes()$ of "ThemeFinder" contains a heuristic implementation of Def. 6. Now the difference between a topic/label and a theme is clear. Parts of a topic can become a theme if they are present for m periods.

To reduce the numbers of automatic created feature spaces and the adjustments of these, the algorithm "ThemeFinder" may still use the old feature space (build in the previous or some earlier time period) if that feature space leads to a model that satisfies the quality requirements of the application. This quality requirements are described in the following definitions. If the old feature space does not satisfy the quality requirements, then it is not useful for the data of the actual period and we have to start the analysing process again using "ThemeFinder" with the feature space of the actual period.

We term a clustering as result of the clustering process that satisfies the quality requirements a "good clustering" and define it as follows:

Definition 7 (Good Clustering) *The document archive is D. fs is the used feature space and $\zeta(D, fs)$ is the clustering of D under the usage of fs. The clustering will be a* good *clustering if the number of labelled clusters of Def.: 4 in it is no less than a threshold $\tau_{clustering}$.*

The threshold $\tau_{clustering}$ adjust the number of labelled clusters of Def. 4 that the clustering is called a "good clustering". Depending on the selected clustering algorithm it can be useful to set the threshold $\tau_{clustering}$ smaller than the number of clusters. Because many clustering algorithms, like the well-known k-means algorithm, have the property to collect all dissimilar data points into one bucket cluster and it will be difficult to get a non-empty label for such a bucket cluster.

The first quality check of the "ThemeFinder" is to control if the usage of the old feature space leads to a "good clustering" after our Def. 7.

The second quality check at "ThemeFinder" will be to check if the used feature space of a previous period is also a good feature space for the actual period. For the decision

on changing the feature space, ”ThemeFinder” relies on the following notion of ”good feature space”:

Definition 8 (Good Feature Space) *Let D_i be the document set of period t_i, fs be a feature space and $\zeta(D_i, fs)$ be the clustering of D_i using fs. The feature space fs is termed as a* good feature space *if (a) $fs \neq FS_i$, (b) $\zeta(D_i, fs)$ is a good clustering and (c) the number of labels of labelled clusters from $\zeta(D_{i-1}, fs)$ that were again found among the labels of labelled clusters in $\zeta(D_i, fs)$ is not less than a threshold $\tau_{matches}$.*

According to this definition, a feature space is good for the document set D_i if it results in a good clustering *and* retains most of the labels found in the previous period(s). In this definition and in Def. 7, it is implicitly assumed, that the period-specific feature space always delivers a good clustering of the document set, since it reflects exactly the dominant features of the documents. If the feature space of the previous period is also a good feature space for the actual period according to the Def. 8, the feature space for the actual period needs not be generated; thus reducing the processing overhead.

We have described the definitions as the basis of our framework. In the following we explain the complete framework and the included algorithm ”ThemeFinder” for label monitoring in detail.

3.3 The Framework

The ”ThemeFinder” operates on a stream of documents monitored at time periods. At each time period, the document set D_i inside the window is first clustered and labelled with the feature space used thus far. The ”ThemeFinder” decides whether the result is acceptable (according to Def. 7 and Def. 8). If it is not the case, the feature space is recomputed and clustering and labelling is re-done. Finally the labels of the labelled clusters in the previous clustering are compared to those of the labelled clusters in the current clustering and persistent themes, as well as label changes are reported.

In Figure 3.3 we present the complete framework as an UML activity diagram to show the individual steps at a document stream to monitor labels over time. The steps of the framework in the coloured rectangle are the steps from the ”ThemeFinder”. The steps before are pre-steps for the ”ThemeFinder” with different standard methods as part of framework.

We explain these steps in the following:

Partition the Time Axis. The documents of the archive come to the archive as a stream. This means that each document is associated to a time point - the time it was added to the archive. For the monitoring of such a document archive, in which new documents arrive as a document stream, it must be decided how to partition the time axis. This depends usually on the application and its goals of the monitoring process. The specification of the time partitions obviously influences the results of the monitoring

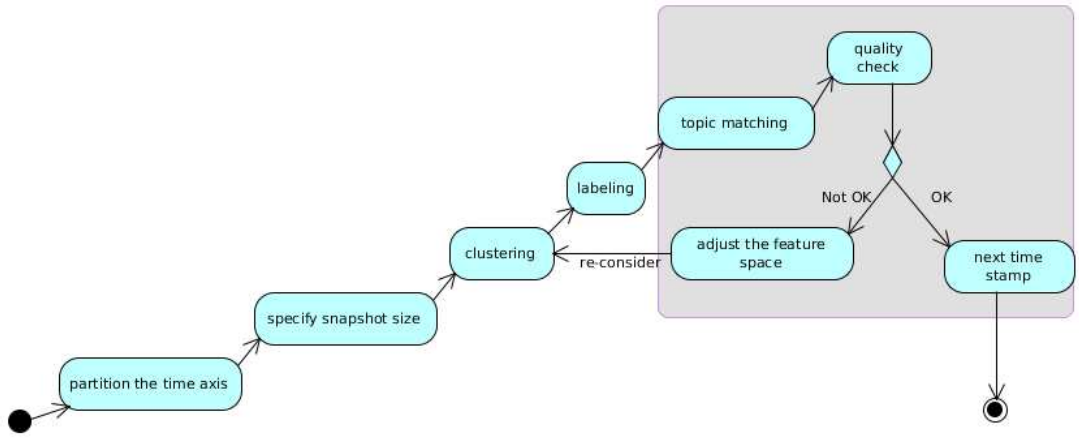


Figure 3.3: framework

process. Here this issue is not more investigated but rather assumed that the application owner has specified the time periods at which the monitoring should take place, according to the objects of the monitoring process. For example, a library may monitor the topics of arriving scientific literature once a year, while a news agency may monitor the topics once a week or even once a day.

In our experiments in chapter 5 we have partitioned the stream into yearly partitions.

Specify the Snapshot Size. After partitioning the time axis it has to be selected a forgetting strategy, especially for one of the two datasets which we have introduced at the beginning of chapter 3.2. There we introduced the accumulated and the non-accumulated dataset as two different cases for a forgetting strategy.

In our experiments we show results with the first two strategies, the accumulated and the non-accumulated dataset.

Clustering and Labelling. After the definition of the document set with the partitions and the forgetting strategy it can be started with the clustering and labelling process as explained before in this chapter. This includes on the one hand the typical data preparation steps at documents like stemming and stopword removing and on the other hand the feature space creation and the vectorisation of the document set. The vectorised document set is clustered with a selected clustering algorithm and then the labelling of the clusters will be done.

After the clustering of the document sets at the different time periods the monitoring process is started. For this process we have developed the algorithm ”ThemeFinder” which is used for the steps ”topic matching” and ”quality check” in our framework. The algorithm is explained in detail in the following sub-chapter.

Adjust Feature Space. We differentiate between one period at which the feature space of the actual period is used on the one hand and on the other hand the feature space of the previous period is used. Because the good feature space of the previous period is also a good feature space at the actual period according to Def.: 8. The usage of the feature space of the actual period could have two reasons, it is the starting period or the feature space, used at the period before as good feature space is not a good feature space at this period as defined before at Def. 8.

In our framework we have not specified the specific clustering algorithm or the specific labelling method. Depending on the document set different algorithms produce better clustering and labelling results. We have only specified that for the complete monitoring process with the framework the selected clustering algorithm and the selected labelling method should be constant, so that the produced labels are comparable with each other. The non-specification of the clustering algorithm and labelling method gives the user of the framework the freedom of selection of both depending on the application and the document archive.

During the experiments we will show that the framework works well at different clustering algorithms and different labelling methods.

3.3.1 Algorithm "ThemeFinder"

In table 3.1 the pseudo-code of the algorithm "ThemeFinder" is shown as basis for our framework shown before in chapter 3.3. We will explain it step by step.

From line 1 to 3 we specify the feature space, the clustering with this feature space and the label set of the labelled clusters at this clustering (Def. 4) from the time period where we start our observation of the document collection over time.

The iteration over all other time periods we start in the loop at line 5. The first quality check is done at line 7; there the number of labelled clusters must be over the threshold $\tau_{clustering}$ to be a good clustering according Def. 7. If this check is ok, for each non-empty label of the clustering of the time period before the function *best_match()* will return the corresponding label to it at the clustering of the time period i (line 11-12).

In line 14 we increment the counter "matches" for the identified corresponding labels. After the matching we check in line 15 and 16 according to Def. 6 if the found label is a candidate to be a theme. As last quality check we examine if the number of corresponding (recovered) labels are equal or better to the threshold $\tau_{matches}$ (line 20). If that is not the case the used feature space will be replaced by the new automatic created feature space of the actual period and the clustering will rebuild at line 22 and 23.

The identified corresponding labels to one label will be unioned with the set of the other corresponding labels of the other labels to get the set of all corresponding labels up to this point in time (line 26).

At last step of the algorithm to find themes as defined in Def. 6, the "ThemeFinder" uses the method $extract\_themes()$ and the threshold m to extract the themes from the set of corresponding labels (line 28).

After identifying the corresponding labels by our algorithm "ThemeFinder" all labels of the time period t_{i-1} for which no corresponding label at the set of labels of period t_i

was found, can be marked as dead labels. All labels of time period t_i, which are not a corresponding label to the labels of time period t_{i-1} are called "new" or "born" labels.

Step	Action
1	$fs \leftarrow FS_1$; $fs_{orig} \leftarrow fs$
2	$\zeta_1 \leftarrow \zeta(D_1, fs)$
3	$L_1 \leftarrow \{label(c) \vert c \in \zeta_i\} - \{e\}$
4	$Collectionthemes \leftarrow L_1$; $subL_1 \leftarrow L_1$
5	**for** $i = 2, \ldots, T$ **do**
6	$\zeta_i \leftarrow \zeta(D_i, fs)$; $subL_i \leftarrow \emptyset$; $matches = 0$
7	**if** $thematic\_clusters(\zeta_i) \geq \tau_{clustering}$
8	**then**
9	$\xi \leftarrow \zeta_i$
10	**for each** $c \in \zeta_{i-1}$ **do**
11	**if** $label(c) == \{e\}$ **then continue**
12	$c' \leftarrow best\_match(c, \xi)$
13	**if** $c' \neq \emptyset$ **then**
14	$matches++$;
15	$l \leftarrow label(c) \cap label(c')$
16	**if** $card(label(c)) - card(l) < \tau_{deviation}$ **then** $subL_i \leftarrow subL_i \cup \{(c, c', l\}$
17	**endif**
18	**endfor**
19	**endif**
20	**if** $matches < \tau_{matches}$
21	**then**
22	$fs_{orig} \leftarrow fs$; $fs \leftarrow FS_i$
23	$\zeta_i \leftarrow \zeta(D_i, fs)$
24	**endif**
25	$L_i \leftarrow \{label(c) \vert c \in \zeta_i\} - \{e\}$
26	$Collectionthemes \leftarrow Collectionthemes \cap L_i$
27	**endfor**
28	$themes \leftarrow extract\_themes(\cup subL_i, m)$

Table 3.1: "ThemeFinder" for label monitoring

Now we describe in detail the method $best\_match()$. In this function the labelling matching process to find the corresponding labels is implemented.

3.3.2 Match Labels with $best\_match()$

The actual finding of the corresponding labels of a clustering to a given label will be done with the method $best\_match()$. This method has two input parameters. The first parameter is a label L_C of a cluster $C \in \zeta(D_i, fs)$ of the period to which the corresponding label

will be found. The second parameter is the set of labels from the clustering $\zeta(D_{i\pm1}, fs)$ of the time period in which the corresponding labels are searched. During "normal" usage, which is from past to present at time axis, the first label is taken from period t_{i-1} and the set of labels is taken from time period t_i. Also a backward approach is possible with this algorithm.

The pseudo code of the method $bestMatch()$ is shown at table 3.2. The line numbers at the following explanations refer to this code. The input label L_C is compared to all labels L_X of the input clustering (line 2-9). The comparison of two labels L_C with label L_X can have three different results.

1. $label(C) == label(X) \rightarrow$ the labels are identical
2. $label(C) \cap label(X) \neq \emptyset \rightarrow$ both labels have a non empty union but are not identical
3. $label(C) \cap label(X) == \emptyset \rightarrow$ the labels have no common word

In the first case the corresponding label is found and will be returned by the method. In the third case, the labels have no common word and the method select the next label L_X from the set of labels to compare with the label L_C. In the second case it exists the possibility to find more than one label L_X in the set of labels which have common words with the input label L_C. All labels L_X, which have common words with the label L_C, are collected in a candidates list (line 7). If all labels of the set of labels are compared with the input label L_C, it will be checked if the list of candidates is not empty. If the list of candidates has only one list item, it will returned as corresponding label (line 13).

Otherwise the labels at this list will be ordered according to the statistics of the words at the labels, starting with the word with the highest statistic (line 14). Now the algorithm picks up each label L_X from the candidate list in which words are appearing in L_C and have similar statistics. Frequent words take precedence (line 16). This means, that the words have a similar importance for the label. If it exists still more than one label after this step as good candidate, then the number of common words at the labels L_C and the candidates will be calculated (line 21). The list of candidates will be sorted on descending order of the frequency of shared words. The label with the highest number of common words will be selected and returned as the corresponding label to the label L_C .

Summary We presented our core idea to handle document streams over time to monitoring the labels of the document clusters over time. Before we described the developed framework in detail, we introduced the needed definitions to understand the new framework and the following algorithms. The presented framework gives a detailed overview about all steps to handle document streams under the goal of monitor the document cluster labels over the time periods. Specially we described the "ThemeFinder" which find corresponding cluster labels from one time period to an other time period. The main method *best_match* is introduced including the pseudo-code.

Step	Action in $best\_match(C, \xi)$
1	$candidates \leftarrow \emptyset$
2	**for each** $X \in \xi$ **do**
3	**if** $label(X) == label(C)$ **then**
4	**return** X
5	**endif**
6	**if** $label(X) \cap label(C) \neq \{e\}$ **then**
7	$candidates \leftarrow candidates \cup \{X\}$
8	**endif**
9	**endfor**
10	**if** $candidates == \emptyset$ **then**
11	**return** $\emptyset$
12	**endif**
13	**if** $len(candidates) == 1$ **then return** $candidate\_label$
14	$L \leftarrow ordering(label(C), MFWF)$
15	**for each** $w \in L$ **do**
16	$wL \leftarrow \{X \in candidates \mid w \in label(X) \& support(w, X) \approx support(w, C)\}$
17	**if** $wL \neq \emptyset$ **then**
18	$candidates \leftarrow wL$
19	**endif**
20	**endfor**
21	$L \leftarrow ordering(candidates, MCWF)$
22	**return** $firstOf(L)$
	$MFWF = Most\_Frequent\_Word\_First$
	$MCWF = Most\_Common\_Words\_First$

Table 3.2: The method *best_match*

4 Experiments

After the detailed definition and description of the ”ThemeFinder” we can start with the experiments. The objective of our experiments is to study the functionality of ”ThemeFinder” under the assumption, that we find topics at the used dataset which can be monitored over time. We used a sub-collection of the publicly available ACM library as dataset, which is a collection of research papers over many years.

We have used this dataset because it is publicly available, has an existing hierarchy by keywords of the papers, which can be used as ground truth and is covers a large time span. Another reason is that this dataset is about a field of computer science that has experienced changes over time, so that it supports our assumption to expect monitorable topics at this dataset.

We conduct experiments (1) when data is accumulated over time and no documents are removed and (2) when a window slides over the data and the oldest ones are left out.

The first group of experiments we have conducted to study the impact of different thresholds at the ”ThemeFinder”, are, for example, number of clusters or number of refound labels. So we get some first understanding of the dataset and assess how many topics it contains at a given point in time.

After studying the impact of the different input parameters and thresholds, we explore the influence of the clustering used and labelling methods to the monitoring process with ”ThemeFinder”.

As last experiments we compare our monitoring results with the ”ThemeFinder” with monitoring results at the same dataset with the well known FOCUS framework from Ganti et al [Ganti 99b] to evaluate the quality of our monitoring process.

4.1 The ACM sub-archive

Before we describe each experiment in detail, we describe the used dataset.

For our experiments we need a text archive with different points in time. We decide to use the H.2.8 ”database applications” sub-archive of the ACM archive [1].

This dataset is a collection of publications by this organisation and is organised by an hierarchical cataloque. This catalogue can be used as classification of each publication and so we can use it as a ground truth to evaluate our experimental results. For evaluation we used on the one hand the labels of the clusters and tried to manually match each label found to a category of the original categorisation of the ACM archive. On the other hand we used the dominant original category there the most documents in a cluster

[1] http://portal.acm.org/ccs.cfm

originate from and compare it to our founded labels and check if the labels predict the same category.

Once made a paper available to the users, the paper is categorised into the ACM hierarchy by using the keywords which stem from the ACM conceptual hierarchy. The sub-archive ”database applications” has 5 subgroups. The group “image databases” appears already in the first period of observations ($\leq$ 1994), the group “data mining” first appears in 1995, “spatial databases and GIS” in 1996, while “scientific databases” and “statistical databases” are used since 1997. The ACM categories are listed in Table 4.1, together with the acronyms we have assigned to them for brevity.

Data mining	DM
Spatial databases & GIS	SpatDB
Image databases	ImgDB
Statistical Databases	StatDB
Scientific Databases	SciDB

Table 4.1: The ACM categories in section H2.8

We downloaded all publications at this sub-archive as html-files. After this, we parsed those html-files to identify the sub-archive name (if it exist), to which the publication is classified in the ACM hierarchy, the year of publication, the title and the keywords. We considered the title and keywords of each document, removing the ACM sub-archive name to which the document was assigned. We did not consider abstracts, because many documents did not have an abstract and those having one could otherwise bias the feature space contents.

After downloading and parsing the relevant informations we have grouped all documents by year of publication. So we have documents at different time periods ranging from $\leq$ 1994 until 2004.

Following our framework, first we have to preprocess the documents so that they are ready for usage with the clustering algorithms. We preprocessed the document excerpts with the tool “DIAsDEM Workbench” [Graubitz 01], which offers basic NLP preprocessing and stopword removal and vectorisation with TF$\times$IDF weighting. The DIAsDEM Workbench is a text mining algorithm for cluster discovery and labelling. We used its k-means and bisecting k-means clustering algorithm with Euclidean distance but replaced its cluster quality evaluation mechanism [Winkler 02] with our notion of thematic cluster and cluster label, setting $\tau_{wordsupport}$ to 0.60. At the first experiment we show clustering results with different numbers of clusters.

At this step we have all documents of the sub-archive ”database applications” grouped Our goal of the experiments is to see temporal themes and persistent themes over time, this is why we have derived two different datasets from this data, depending on the forgetting strategy. On the one hand, we use this dataset with total forgetting of previous time periods to see temporal themes or hypes at the data and call it the dataset 1. On the other hand we remember all data points and have no forgetting over time to see persistent

themes over the observed time span. This dataset we name dataset 2.

The complete forgetting strategy means that each time period only consists of the documents which are published at this period. This is the forgetting strategy with a windows size of the value one. The distribution of this documents at each point in time is presented in table 4.2. Note, that this dataset have include only documents till august 2004. This dataset is our dataset 1, which is a non-accumulated dataset without any sub-archive informations.

Period	$\leq$ 1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	01-08/2004
numbers	1334	144	183	313	377	495	653	518	716	986	173

Table 4.2: Number of documents in the ACM sub-archive “database applications”

The second data set (dataset 2) has no forgetting. But we wanted also to use the sub-archive informations of this dataset and so we downloaded this sub-archive again, this includes some documents at the last time periods more as at the first download. The documents in this dataset, which are not a member of any sub-archive and so are only inserted into the main group ”database applications”, we have not used at this dataset. The reason for this is that we will use the categorisation into the subgroups of the original archive by the ACM as a ground truth for an evaluation of our experiments with this dataset.

Caused by the different first appearance of documents at the subgroups we decide to use only documents from 1996 until 2004. The distribution of documents is shown in Table 4.3 as dataset 2 without forgetting, this means all documents of previous periods are also present at the actual period. This is our dataset 2 for the experiments in the following sub-chapters.

Period	1996	1997	1998	1999	2000	2001	2002	2003	2004
numbers	89	150	369	675	1155	1634	2338	3371	4434
DM	16	56	148	315	580	872	1330	1984	2577
SpatDB	40	53	124	188	316	388	517	662	851
ImgDB	16	22	70	135	208	287	340	429	571
StatDB	17	19	21	25	33	44	66	84	89
SciDB	0	0	6	12	18	43	85	212	346

Table 4.3: Number of documents in the subgroups of the ACM sub-archive “database applications”

Now we have two datasets, a dataset with forgetting over time, dataset 1 and a dataset 2, which has no forgetting over the time. We start with experiments on both datasets to test the functionality of our algorithm ”ThemeFinder”.

4.2 Impact of input parameters

We make the first group of experiments to study the impact of the input parameters to the ”ThemeFinder”. We will find out if different values of the thresholds and input parameters have an influence to the monitoring process and to the quality with the ”ThemeFinder” under the main goal of ”ThemeFinder” to find and monitor themes over time. This includes the experimental check if our framework is able to monitor existing labels of the clusters over time.

We used the dataset 1 because it is more difficult to find short present hypes as themes at the dataset as to find persistent themes over the whole time span. We used the DIAsDEM tool for the preprocessing steps and vectorised the dataset with a feature space (Def.1), where we used only the 30 dominant words of the dataset. As clustering algorithm we used the bisecting k-means clustering algorithm because this algorithm is an easy to use and well known clustering algorithm from the literature and this algorithm produce better results as the standard k-means algorithm [Karypis 00b].

In the following sub-chapters we study the behaviour of ”ThemeFinder” when varying the number of clusters. Then we experiment with different thresholds values for $\tau_{matches}$ and $\tau_{thematic}$ (cf. Table 3.1, line 7). As a closing step for this first experiment part we analyse the dataset again with the best values for cluster number and threshold values.

4.2.1 Impact of number of clusters

The number of themes, that our algorithm can find, depends on the number of thematic clusters it finds in each period of observation. Hence, we have varied the value of k for bisecting k-means because we suggest that we can find a value for cluster number which is useful for the monitoring process. Since bisecting k-means generates one bucket-cluster, in which all otherwise dissimilar vectors are put together, a clustering can contain at most $k - 1$ thematic clusters. Accordingly, we have set the threshold $\tau_{thematic}$ (cf. Table 3.1, line 7) to $k - 1$ and $\tau_{matches} = \tau_{thematic} - 1$, thus requiring that a clustering over a given feature space is good if all except the bucket-cluster are thematic clusters. As explained in Table 3.1, if a clustering contains less than $\tau_{thematic}$ thematic clusters, then the feature space is replaced by the period-specific feature space.

The number of thematic clusters monitored by ”ThemeFinder” at different input values for the number of clusters (k) is shown at Figure 4.1.

The horizontal axis in Figure 4.1 shows the observation periods, the vertical axis counts the thematic clusters found in each period. We start with a relative small number of clusters, because of the small number of documents at the different time periods. It must be stressed that the feature space is not always the same. If a feature space is not good enough after the quality check of thematic clusters in comparison with the threshold $\tau_{thematic}$, the feature space is replaced with the period-specific feature space following our framework descriptions. This is indicated by a downward peak in the curve, as it is the case of $k = 7$. If the period-specific feature space cannot deliver adequate thematic clusters either, then the curve stops, as it is the case for $k = 6$ and $k = 4$.

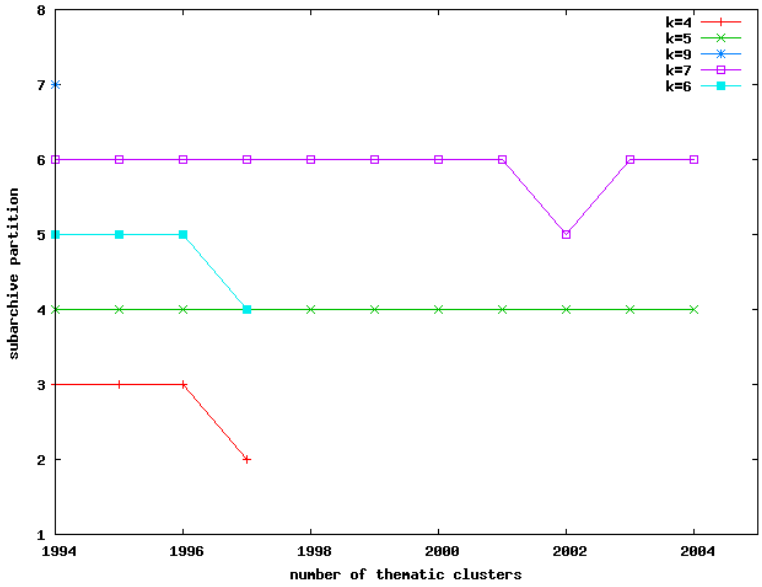


Figure 4.1: Number of thematic clusters for different values of k

If a curve is a horizontal line (e.q. curve for $k = 5$), then an adequate number of thematic clusters was found in the clustering of each period. Then, k is an indicator of the number of themes that are in the sub-archive. It must be noted that the clusterings were not necessarily built over the same feature space: According to Table 3.1, the feature space may be replaced if the clustering ζ_i does not have sufficient matches to the previous clustering ζ_{i-1}.

A break in a curve (cf. curve for $k = 7$ at period 2002) indicates that a clustering delivered less than thematic clusters than $\tau_{thematic}$, thus triggering a replacement of the current feature space with the period-specific one. For $k = 7$, the new feature space has delivered good clusterings for the next periods. For $k = 6$ and $k = 4$, this was not the case: Both curves stop at period 1997, at which not even the period-specific feature space could deliver a sufficient number of thematic clusters.

As result of this experiment we find out the value $k = 5$ can be a good selection for the input parameter to the bisecting k-means algorithm.

As another result we see also that not for all values of k and the periodic-specific (actual) feature space enough thematic clusters can be found.

As next step we find out the impact of the thresholds as input parameters to the ”ThemeFinder”, like $\tau_{thematic}$ or $\tau_{matches}$.

4.2.2 Impact of thresholds

The new ACM topics in the sub-archive indicates that the ACM taxonomy designers have responded to emerging research threads. These threads are associated with a drift in the frequent terms in the documents, new research areas use new terms. A simple way

of detecting such a drift is by clustering the documents and check whether the thematic clusters degenerate. So, we first checked whether the anticipated themes could be found without using "ThemeFinder".

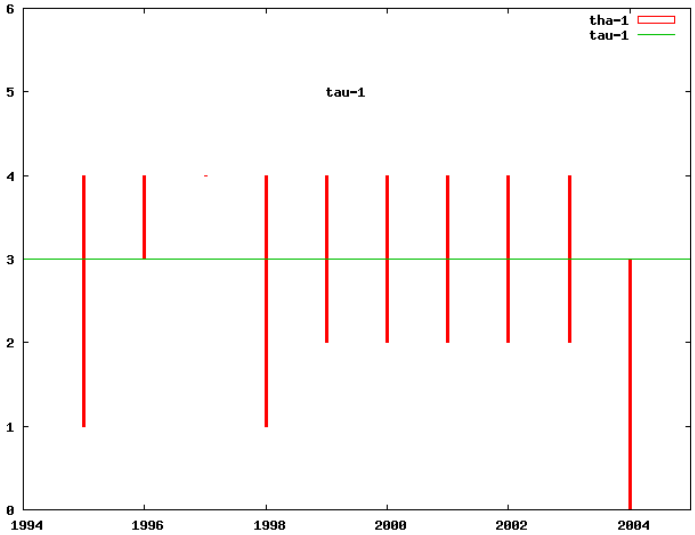
To this purpose, we clustered the documents of the first period with the corresponding feature space using $k = 5$ (cf. experiment before Chapter 4.2.1) and $\tau_{thematic} = k - 1$. Then, we gradually assigned the documents of the subsequent periods to the clusters. At each period, we checked (a) for changes in the distribution of documents in the clusters and (b) for drastic changes in the cluster labels, e.g. disappearance of words from a label, occurrence of new words or large disparities in the support of a word in a label. For cluster matching, we used the heuristics in $best\_match(\cdot)$ in Table 3.2. When changes occurred, re-clustering with the period-specific feature space was performed. It turned out that re-clustering was needed at each period, whereupon the period-specific feature spaces gave raise to short-lived themes. Hence, this simplistic method did detect drifts but not themes. Thus, we have run "ThemeFinder" to identify stable clusters and their labels.

The goal is to find out the impact of the threshold $\tau_{matched}$ to the monitoring process under the main goal of finding and monitoring as many themes as possible. We used the same constraints of $k = 5$ and $\tau_{thematic} = k - 1$, i.e. we required at least 4 thematic clusters. Then, we varied the value of the threshold $\tau_{matches}$ that determines the minimum number of thematic clusters that should survive in the next period. If this threshold is violated, the feature space is given up and the period-specific feature space is used instead (cf. step 20 of "ThemeFinder" in Table 3.1). We have experimented with $\tau_{matches} = \tau_{thematic} - i$ with $i = 1, 2, 3$ and the results are shown on Figure 4.2.

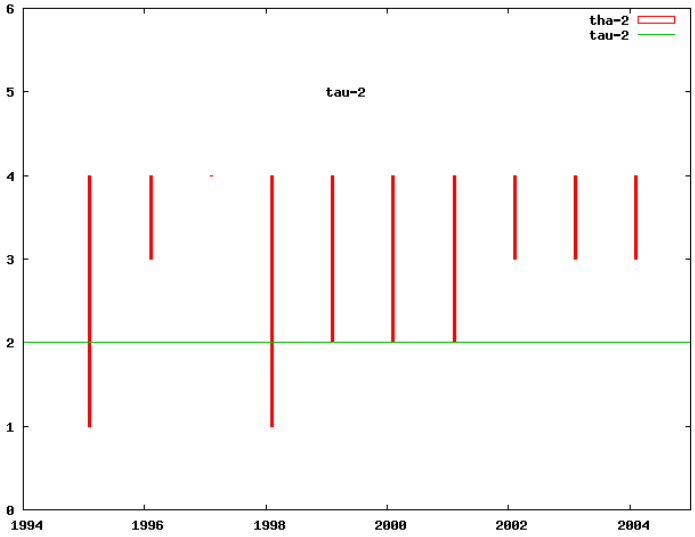
In Figure 4.2, the horizontal axis depicts the periods of observation, while the vertical axis refers the thematic clusters in each period. The horizontal line drawn at the value of $\tau_{matches}$ is the baseline. For each period of observation, we see the number of thematic clusters found, the top point of the vertical line, and the number of matched clusters amongst them, bottom value at the vertical line. The number of thematic clusters should be at the value 4 (k-1), otherwise a re-clustering of the period with the periodic-specific dataset will be done. The vertical line segment at each period is the difference between these two values. It means that a shorter vertical line is better because more of the thematic clusters are in the set of matched clusters too. If this segment crosses the baseline, then the number of matches is less than the threshold $\tau_{matched}$, hence triggering a re-clustering with the period-specific feature space.

As expected, smaller values of the $\tau_{matches}$ threshold result in less adjustments of the feature space. For $\tau_{matches} = \tau_{thematic} - 1$, see Figure 4.2(a), it can easy be seen by counting the crossing vertical line with the horizontal line, that the feature space must be changed in 8 out of 10 periods (the first period is not counted, obviously). Of these changes, 7 are due to the value of $\tau_{matches}$ and one to $\tau_{thematic}$ itself. In 2004, the number of thematic clusters was already too low, so that no cluster matching was performed and re-clustering with the period-specific feature space was triggered.

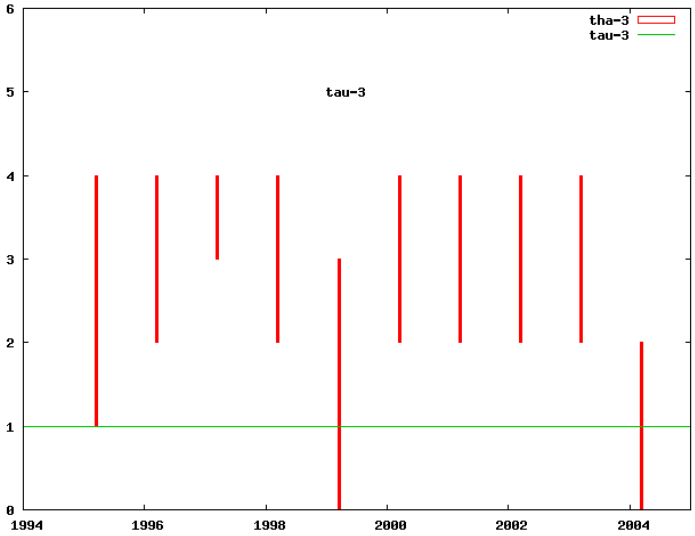
A high number of feature space changes is not desirable, because it is apt to features of short-term popularity and prohibits a long-term observation of the clusters. In addition, each feature space creation at a business project is time and cost intensive, because



(a) $i = 1$



(b) $i = 2$



(c) $i = 3$

Figure 4.2: Thematic labels for $\tau_{matches} = \tau_{thematic} - i$ with $i = 1, 2, 3$

an automatic generated feature space has many dimensions and increases the time to vectorise and cluster the documents which are represented by such a feature space. For $\tau_{matches} = \tau_{thematic} - 2$, a change in the feature space is needed only for 2 periods, seen at Figure 4.2(b). The same holds true for $\tau_{matches} = \tau_{thematic} - 3$, see Figure 4.2(c), which is less restrictive. Although the value of $\tau_{thematic}$ is too small (4 thematic clusters) for generalisation, this experiment indicates that the value $\tau_{thematic} - 2$ is appropriate for $\tau_{matches}$.

With this experiment we have seen the impact of the threshold $\tau_{matched}$ to the monitoring process with "ThemeFinder" over the time. We see that if the threshold is very restrictive, we increase the number of feature space adjustment dramatically. Although we have a relatively small number of clusters and matched clusters, we can monitor the existing clusters over time with "ThemeFinder". Now we have values for all input values to our framework and can start with the real experiment; the usage of our "ThemeFinder" at the dataset 1. This is the dataset with the complete forgetting strategy, in order to find temporary hypes of themes at the dataset. It must be noted, of course, that we have already used our framework at both previous experiments, but the goal was not the functionality check of our framework. It was only to study the impact of the input values and get more familiar with this dataset.

4.3 "ThemeFinder" on dataset 1 when all past data are forgotten

To run the complete monitoring process with our framework and our "ThemeFinder", at both previous experiments we have studied the impact of the different input values for the clustering algorithm and "ThemeFinder" used, the number of clusters (k) and the threshold values for thematic and matched clusters. One effect of studying this impact is that we know now a good selection of those input values for further analysis on dataset 1. The following short Table 4.3 summarises the input values to the framework process, from vectorisation over clustering to monitoring the cluster labels with the "ThemeFinder":

dataset	feature space	cluster numbers	$\tau_{matches}$	$\tau_{thematic}$
dataset 1	top-30	5	2	4

Table 4.4: inputs for experiment with dataset 1

In this experiment we assume to find hypes of themes at different time periods and perhaps also some themes which are present over more time periods. We make this assumption because we use the complete forgetting strategy and so as a basis each time period has nothing in common with the other time periods. At this point it is more complicated to find themes which are present over more than the selected time period. We juxtaposed the 6 ACM topics ("database applications" and the 5 subgroups) and sub-labels that were found by "ThemeFinder" at this experiment. No label qualified as "persistent theme" (cf. Def. 5). For the weaker definition of theme, see Def. 6, we

show the impact of different values of the threshold m. In Table 4.5, we present the labels of the thematic clusters in each period. The second column indicates the period whose feature space was used for clustering. For period 1995 and period 1998 we have to update the feature space to the actual feature space. The feature space of period 1998 is used for all later time periods at this dataset because it is a good feature space after our definitions (cf. Def. 8). For each word in a label, we see next to it its support in the cluster; if no support value is shown, then it was 100% accurate. The rightmost column maps the cluster labels to the ACM topics. The mapping was done by calculate the percentage of documents of each subgroup in the cluster. This mapping was done, if this percentage of one subgroup is over 50%. For instance at time period 1997 the first cluster has no dominant group. This mapping could be seen as a ground truth at the clusters at the different points in time.

Empirical evaluation at this experiment is done by an empirical comparison of the real dominant categories of a cluster, the ground truth, and the found labels at this clusters. This comparison is done manually because the words of the original category names must not be automatically dominant words in the existing clusters. But this mapping could be done by means of the labels and the category names from a person with good knowledge in the used field of computer science, the ”database applications” field.

The emerging themes which are monitored by the ”ThemeFinder” are as follows:

- The cluster label {association,mine,rule} appears in the last 5 periods, thus indicating that *“association rules mining”* is a theme, if the threshold is set to $m = 5$. In fact, the sub-label {association, rule} is present for the last 6 periods.

- The label {image, retrieval} is present in only two periods, but appears as a subset of some longer cluster labels in 8 periods, so it would be a theme for any $m \leq 8$. Some clusters associated with this theme also cover “content” (content management?), “base” (image bases?, image databases?) and/or “model”.
  This theme is obviously a derivative of the ACM topic “image databases” existing since 1978, indicating a shift of research towards image retrieval.

- The label {spatial} appears in two periods only, 1998 and 2002. However, the correlated label {gis} appears in 1999. This indicates that a further theme exists, associated with spatial or (more specifically) geographical information systems.

- “Mine” (for: mining) appears as label or sub-label in 4 periods, in 2002 associated with the “web” (web mining?) and in 2003 associated with “decision” (mining for decision support?).

- “Knowledge discovery” is present as label or sub-label in two periods, while “knowledge” appears in labels of two further periods. Nowadays, we tend to observe knowledge discovery as a wide area that subsumes association rules mining; in the sub-archive however, the two labels cover different clusters.

Some early labels or individual words are non-conclusive, such as “design”, “application”, “model”, “information” and “system”. Since we concentrate on the “database applications” sub-archive, we cannot trace the migration of ACM topics to other sub-archives. However, it is likely that articles on {database, model} or “object” (object-oriented models?, object-oriented databases?) are not observed as “database applications” in the recent years.

For the mapping of cluster labels to ACM topics, we did not consider document contents. For this, one should study the content of each document in each cluster and assign it to the appropriate ACM topic, even if the topic did not exists by the time the document was inserted to the archive. Nonetheless, the juxtaposition of labels to ACM topics already reveals interesting insights, as can be seen from the last column of Table 4.5:

- The ACM topic “image databases” is clearly associated with the theme “image retrieval” and is mapped to the clusters having this theme as label or sub-label.
- The ACM topic “data mining” covers two separate and persistent clusters, one on the theme “association rules mining” and one on “knowledge discovery”.
- The ACM topic “spatial databases” covers the labels “spatial” and “gis”.

The ACM topics “scientific databases” and “statistical databases” have no associated cluster labels, nor themes. One likely explanation is that the expression “scientific database” (resp. statistical database) describes a large group of database types and application types without dominant common terms. Furthermore, some research articles on these ACM topics are likely to adhere to other themes, as well: Knowledge discovery is a likely subject in papers on statistical databases, while some papers on scientific databases may refer to image retrieval (e.g. of medical or astronomical images). Another explanation can be the small number of documents in both subgroups, see Table 4.3, in comparison to the dominant subgroup ”data mining” or the both other groups ”image databases” and ”spatial databases”.

As result of this experiment we see that the ”ThemeFinder” find themes at the periods which are present more than only one period although many labels are only present at one period. Hence, despite the discrepancy between the number of themes and the number of ACM topics, the ”ThemeFinder” categorises the sub-archive’s content in a reasonable way: It associates documents with emerging themes, allows for a mapping with ACM topics and gives indication about the coverage of these topics on the sub-archive.

period	*feature space of period*	*label* $\tau_{wordsupport} \geq 0,6$	*category of ACM*
1994	1994	technology information(0,625); retrieval (0,666); system (0,645) design system	
1995	1995	base, database, image, retrieval database(0,8), object, system(0,8) database(0,8), model(0,7) information	image databases scientific databases database appl. database appl.
1996	1995	base, database, image, knowledge, model application system database	image db database appl. database appl. database appl.
1997	1995	application knowledge(0,83) information(0,71), system(0,79) database(0,92)	 data mining database appl. database appl.
1998	1998	base(0,77), content(0,85), database(0,77), image(0,62), retrieval spatial discovery(0,83), knowledge(0,91), mine(0,83) information	image db spatial db + GIS data mining database appl.
1999	1998	base(0,65), content(0,65), database(0,61), image(0,91), retrieval(0,96) gis(0,96) association(0,73), rule mine(0,62)	image db spatial db + GIS data mining data mining
2000	1998	image(0,75), retrieval(0,81) association(0,97), mine(0,69), rule(0,97) discovery, knowledge(0,86) information	image db data mining data mining database appl.
2001	1998	design association(0,9), mine(0,9), rule base(0,75), image(0,92), retrieval(0,97) mine	database appl. data mining image db data mining
2002	1998	mine(0,92), web base(0,9), content(0,67), image(0,76), retrieval(0,9) association(0,96), mine(0,79), rule(0,88) spatial	data mining image db data mining spatical db + GIS
2003	1998	image(0,8), retrieval(0,95) decision, mine(0,7) association(0,96), mine(0,81), rule(0,9) information(0,99)	image db data mining data mining database appl.
2004	1998	base(0,91), image(0,65), retrieval(0,7) association, mine(0,93), rule mine(0,63) database(0,82)	image db data mining data mining database appl.

Table 4.5: labels and used feature spaces at dataset 1 with ”ThemeFinder”

4.4 ”ThemeFinder” on dataset 2 when no data are forgetting

In contrast to the experiment before we use the dataset 2 here, which has as forgetting strategy to use all documents and forget no document over the time.

The target of this experiment with the no forgetting strategy is to find persistent themes over the whole time span or themes which are present at many time periods through time-based analysis of this archive. Our assumption is that persistent themes should be similar

to the given ACM category names of the used sub-archive. But it is also possible to find persistent themes which are subtopics of the given ACM taxonomy or synonyms of this. Both gives the opportunity to adjust the existing taxonomy. In particular, we are interested in (i) identifying the themes in the sub-archive, (ii) juxtaposing the themes found by our framework to the 5 ACM subcategories and (iii) studying the impact of the different thresholds for define a theme on the performance of "ThemeFinder".

We used the dataset 2, again using the tool DIAsDEM to vectorise this dataset and set the feature space again to the 30 most used words. As a result of the experiments in Chapter 4.3 we select as setting $k = 5$ and $\tau_{wordsupport} = 0.6$ with the hope that those initial values are also good selections at the dataset with no forgetting strategy. The input values are summarised in Table 4.4.

dataset	feature space	cluster numbers	$\tau_{matches}$	$\tau_{thematic}$
accumulated	top-30	5	2	4

Table 4.6: input values for experiment with dataset 2

Our founded labels of each period are shown in Table 4.7 and discussed below.

The first column in Table 4.7 shows the time period under observation. In the second column, we see the feature space used by "ThemeFinder" for the clustering. For 1997, the old feature space of 1996 has been replaced by the period-specific feature space. Different from our experiments on dataset 1 with complete forgetting, this feature space has turned out to be adequate for all subsequent periods.

The labels found by our framework are shown in the third column. Next to each word, we see its support inside the cluster. We can see that there is a gap in the support of the words in the label. If $\tau_{wordsupport}$ were set to any value larger than 0.7, only words appearing in all documents would have qualified. This would have lead to shorter labels but also to the disappearance of some thematic clusters, like the cluster labelled "datum" which refers to data mining (this label is discussed below).

Parameters affecting the number of themes discovered: The third column shows that there are no persistent themes according to Def. 5, since no label persists across all periods. However, there are several, quite interesting, themes. When we set the number of periods m to 4 and insist that no word from a label may disappear ($u = 0$), the label {datum, mine} qualifies as theme, while the label {retrieval, image, base} persists in 4 non-consecutive periods. If we allow that a label may change by at most one word ($u = 1$), then {retrieval, image} with the additional word "base" becomes a very stable theme, appearing for the last 5 time periods. This theme refers obviously to "image retrieval", a subcategory of image databases that emerges in 1997, disappears for a short time and then becomes stable from 2000 on.

The emergence and evolution of labels associated to data mining is also very interesting. The first cluster label of period 1996 contains the words "discovery", "knowledge" and "datum" (singular form of data) in all documents, the word "pattern" is also very frequent. With the period-specific feature space of 1997, the cluster on data mining becomes

Time period	*Feature space of*	*Words in the label* $\tau_{wordsupport} \geq 0,6$	*ACM topic name*	*correctness*	*coverage*
1996	1996	discovery (1), knowledge (1), datum (1), gis (0.67), pattern (0.67), spatial (0.67)	DM	0.67	0.25
		. . . COVERS ALSO	SpatDB	0.33	0.5
		database (1), datum (1)	DM	0.53	0.5
		database (1)	–	–	–
1997	1997	datum (1), discovery (1)	DM	0.9	0.5
		image (1), content (1), base (1), retrieval (0.67)	ImgDB	0.83	0.23
		statistical (1), database (1), security (1)	StatDB	0.93	0.68
1998	1997	datum (1), discovery (1), knowledge (1)	DM	0.89	0.26
		datum (1), mining (0.64)	DM	0.9	0.5
		database (1)	–	–	–
1999	1997	datum (1), discovery (1), knowledge (1)	DM	0.92	0.22
		system (1), computer (1)	ImgDB	0.67	0.01
		system (1), geographical (1), information (0.69)	SpatDB	0.9	0.23
2000	1997	datum (1), mine (1)	DM	0.91	0.22
		discovery (1), knowledge (1), datum (0.62)	DM	0.92	0.22
		retrieval (1), image (1), base (0.69)	ImgDB	0.92	0.27
2001	1997	datum (1), mine (1)	DM	0.91	0.21
		datum (1), mining (1)	DM	0.87	0.33
		retrieval (1), image (1), base (1)	ImgDB	0.93	0.36
2002	1997	datum (0.65)	DM	0.69	0.44
		datum (1), mine (1)	DM	0.92	0.23
		retrieval (1), image (1), base (1)	ImgDB	0.91	0.35
		system (1)	–	–	–
2003	1997	datum (0.63)	DM	0.7	0.44
		datum (1), mine (1)	DM	0.92	0.22
		retrieval (1), image (1)	ImgDB	0.91	0.41
		database (1)	–	–	–
2004	1997	datum (0.6)	DM	0.64	0.46
		datum (1), mine (1)	DM	0.92	0.21
		retrieval (1), image (1), base (1)	ImgDB	0.87	0.34
		image (1)	ImgDB	0.78	0.30

Table 4.7: Thematic clusters and corresponding ACM categories for each period at dataset 2

separated under the label {datum, discovery}. The words “knowledge” and “discovery” persist in the next three periods. For $m = 3$, the label {datum, discovery, knowledge} would have become a theme, the “knowledge discovery [from] data”. Starting from 1998, the label {datum, mining} becomes present, the two sibling labels {datum, mine} and {datum, mining} finally absorb the older label {datum, discovery, knowledge} and the new theme for “data mining” becomes a very stable label.

Artefact: mine vs. mining An explanation of the sibling labels {datum, mine} and {datum, mining} is due here. They are an artefact of the linguistic preprocessor, which (correctly) distinguishes between “mining” and “mine”. Since the documents of the ACM sub-archive though are quite unlikely to refer to explosives, we can assume that all appearances of “mine” refer to data mining. For the time being, however, the artefact causes either distinct clusters (as in 2001) or cannibalisation – none of the two words is adequately frequent to appear in a label. We suspect that this is the cause of the uninformative label “datum” that appears in the last three periods. This is further indicated by the juxtaposition of the cluster labelled “datum” to the ACM categories: 64%

of its members refer to data mining. We tried to solve this problem by editing different input files to the tool DIAsDEM at the preprocessing step, but no change has reflect to a solution to this artefact problem.

Evaluation with coverage and correctness: For the fifth and sixth column of Table 4.7 we introduce two measures, emanating from the conventional measures of correctness and coverage for two-class prediction. For any ACM category cat and for any cluster C we define the correctness of the cluster towards the category as the ratio of cluster members belonging to this category:

$$correctness(cat, C) = \frac{|\{x \in C | x \in cat\}|}{|C|}$$

We similarly define the coverage of the cluster towards the category as the ratio of category members that appear in this cluster:

$$coverage(cat, C) = \frac{|\{x \in cat | x \in C\}|}{|cat|}$$

Then, in the last two columns of Table 4.7, we show the correctness and coverage of each cluster C towards its "dominant" category, i.e. the category to which most of its members belong. This corresponds to the category cat with the maximum correctness. We use the value 0.5 for this measure to enforce cluster homogeneity. We see at the correctness column relative high values that show us that the found clusters and labels are nearly from one subcategory of the ACM categorisation. The coverage column has much smaller values that let us assume that many documents of a subcategory at the ACM categorisation are not clearly assigned only to this subcategory because those documents must be collected by the bucket cluster of the bisecting k-means clustering.

Empirical Evaluation: The fourth column of the table shows the dominant category, calculated with the same method as the experiment before with the non-accumulated dataset. For labels like "database" and "system" we did not assess a dominant category. For the other labels we see in the fifth column that the correctness is rather low at first (1996). As soon as the new feature space of 1997 is introduced, though, there is a good mapping of clusters to the individual categories, reaching a correctness of 0.92 for data mining in some periods. For the first cluster in period 1996 we also show the second category present in the cluster. We see that the cluster consists of documents on data mining and on spatial databases in a 2/3 to 1/3 relation.

We can see from Table 4.7 that more than one cluster may be mapped to the same category. This is reflected in the last column, where the coverage towards the dominant category only once exceeds 0.5. This is natural: Categories like DM or ImgDB are very broad and we find some subtopics of this categories. Since we trace only stable themes, the coverage cannot reach 1. This is best reflected in the theme "image retrieval", which is a clear subcategory of image databases. We find no subtopics at the small categories,

the “scientific database” and “statistical database” category.

As result of this experiment we can say that our analysis of the accumulating sub-collection can identify stable *and popular* themes. Some stable themes with lower support, e.g. subtopics of the themes we find here, can be better traced by the analysis of the non-accumulating sub-collections at the experiment before. In comparison to the experiments before we have identified some themes like “association, mine, rule”, a popular sub-area of data mining and we have found the themes “discovery knowledge” and “mine” as two different themes at the non-accumulated sub-collections. Here we have seen both themes as part of an evolution of the main collection theme “datum”.

4.5 Result comparison on dataset with different oblivion strategy

To compare the results of ”ThemeFinder” with the two different forgetting strategy datasets, we have to notice the differences between the targets of both experiments which depends on the specialities of the different forgetting strategies. On the one hand we will find themes which are present over more than one period, so that we can monitor them and will see hypes at the periods. On the other hand, with the no forgetting dataset, we will find persistent themes, it means themes which are present over many time periods or the whole time span.

Similarities between the two oblivion strategies It can be seen that with the given starting values for the input values at both datasets we get with the bisecting k-means clusters with labels, without the bucket cluster. From the monitoring aspect we see that we can monitor the existing labels with the ”ThemeFinder” under the given value 2 for the threshold $\tau_{matched}$. From the result tables for the experiment with the complete forgetting dataset (Table 4.5) and the experiment with the no forgetting dataset (Table 4.7) we see that at both experiments we need to adjust the feature space only 2 or 3 times. At both evaluation results we see that the most clusters originate from the dominant category of the ACM sub-archive, the ”data mining” category.

Differences between the two oblivion strategies A detailed look at the found labels at the result tables 4.5 and 4.7 show that the support of the words at the labels from the dataset with no forgetting is much higher than at the other dataset. Another difference is that the words, which are a member of the labels with the complete forgetting strategy dataset are more concrete as the labels with the no forgetting dataset experiment. One explanation for that can be that the specific words at the labels with the complete forgetting dataset are not so present over the whole time and so the common words from all periods become members of the labels of the clusters.

4.6 Influence of the clustering algorithm on the monitoring results

One advantage of our framework is, that it works on the results of a clustering algorithm, specially only at the labels of the produced clusters. So the idea is that the user of our framework can select the best cluster algorithm to the dataset. That's why the clustering algorithm should not be fundamental for the monitoring process. But with the following experiment we will check the influence of two very different cluster algorithms.

The goal of this experiment is to show that our algorithm "ThemeFinder" works on different clustering algorithm results and that we are able to monitor the produced labels. As dataset we used the dataset 2 without forgetting. As tool for the preprocessing, vectorisation and clustering, we used the Text Clustering Toolkit (TCT) for this experiment from the Trinity College of the University Dublin because they have implemented different quality measures and clustering algorithms.

We decided to use two different basic clustering algorithms. On the one hand we will use a partitioning based algorithm, on the other hand we will use a density based clustering algorithm. The question is, what can be happen to the monitoring results if we change the clustering algorithm? On the one side we decide to use a partitioning clustering algorithm, like k-means or bisecting k-means. On the following we will see that the decision was also to use the bisecting k-means for this comparison experiment. On the other hand we use the Incremental DBScan. One difference between both algorithms is, that the Incremental DBScan produces for each clustering a global optimum, compared to the bisecting k-means. The assumption is that the Incremental DBScan will find labels, but they are only rarely relocatable and therefore persistent over the following time periods. This includes that labels will not died, but new labels arise through the increasing number of documents over time and the constant input parameters (minPts, eps) to this algorithm. We used the incremental version of the DBScan algorithm because this algorithm is designed specially for streams. If we would use the normal DBScan algorithm, we should get the same results if we recluster the existing data set at each end of the time period. So the results can stand for both versions of the DBScan algorithm, the Incremental DBScan and the static one with reclustering of the complete data set at the end of each time period. On the other side we have the bisecting k-means algorithm, which produces no global optimum clustering, but nevertheless we find labels which we can monitor with our "ThemeFinder". So the target at this experiments is to find out the influence of the given clustering algorithm to the monitoring process with the "ThemeFinder".

Partitional Clustering At first we make different experiments with partitioning clustering algorithms. We start to find a good number of partitions of the used dataset 2. Secondly we compare different partitioning cluster algorithms to find a good representative of the partitioning clustering algorithms. The assumption for this step is, that the probability to get good clustering results is much higher if different quality measures agree among each other.

Finding a good partitioning First we start with the bisecting k-means algorithm, as a result from [Karypis 00b] and try to find the best cluster number. Instead of using the results at the experiment 4.2 directly we start to find the cluster number again, because at this step we use the dataset 2 with a new tool and now we have the possibility to compare the results with different quality measures. We produced clusterings with $k = \{2...10\}$ over the whole dataset and calculate different quality indexes. As quality measures we use three different external measures and use the original hierarchy of the ACM library as ground truth. Each of this measures is a representative measure of a subgroup of external measures. We used the Normalised Mutual Information (NMI) [Strehl 02], which is one measure from the group of information theoretic measures. This group has focused on concepts from information theory, which considers the uncertainty of predicting a set of natural classes based on the information provided by a clustering of the same data. As second external measure we used the purity index [Zhao 02], and as third measure the Rand index [Rand 71], which is a pairwise co-assignment measure.

Before we use those three measures, we shortly introduce this external measures.

Excursus on NMI: Formally, let $p'(i)$ and $p(j)$ denote the probabilities, that an object belongs to class C'_i and cluster C_j respectively. Furthermore, let $p(i, j)$ denote the joint probability that an object belongs to both C'_i and C_j. For each data object assigned to a class in C', mutual information evaluates the degree to which knowledge of this assignment reduces the uncertainty regarding the assignment of the object in C. The mean reduction in uncertainty across all objects can be expressed as:

$$I(C', C) = \sum_{i=1}^{k'} \sum_{j=1}^{k} p(i, j) log \frac{p(i, j)}{p'(i), p(j)}$$

$I(C', C)$ takes values between zero and min $(E(C'), E(C))$, where the upper bound is the minimum of the entropy values for the two clusterings. To produce values in the range [0, 1], [Strehl 02] defined normalised mutual information (NMI), where the mutual information between the two clusterings is normalised with respect to the geometric mean of their entropies:

$$NI(C', C) = \frac{I(C', C)}{\sqrt{E(C'), E(C)}}$$

In practice, an approximation for this quantity, based on cluster assignments, can be calculated using:

$$NMI(C', C) = \frac{\sum_{i=1}^{k'} \sum_{j=1}^{k} n_{ij} log(\frac{n * n_{ij}}{n'_i * n_j})}{\sqrt{(\sum_{i=1}^{k'} n'_i log \frac{n'_i}{n})(\sum_{j=1}^{k} n_j log \frac{n_j}{n})}}$$

An accurate clustering should maximise this score, where a value of 1 indicates an exact correspondence between the assignment of objects in C' and C, while a value of 0

indicates that knowledge of C provides no information about the true classes C'. The formula for NMI does have a slight tendency to favour clusterings for larger values of k, although it exhibits no bias against unbalanced cluster sizes.

Excursus on purity index: [Zhao 02] suggested measuring the extent to which each cluster contains objects from a single dominant natural class. The purity of a cluster C_j is defined as the fraction of objects in the cluster that belong to the dominant class contained within that cluster:

$$P(C'_i, Cj) = \frac{1}{n_j} \underset{i}{max}\{N_{ij}\}$$

Unlike the classification accuracy measure, purity allows multiple clusters to be matched to the same dominant class. The overall purity of a clustering is defined as the sum of the individual cluster purities, weighted by the size of each cluster:

$$P(\zeta', \zeta) = \sum_{j=1}^{k} \frac{n_j}{n} P(C', C_j)$$

This measure provides a naive estimate of partition quality, where larger purity values are intended to indicate a better clustering.

Excursus on Rand index: The Rand index count the pairs of objects for which the clusters and natural classes agree on their co-assignment. By considering all pairs, we can calculate statistics for each of four possible cases:

- $a =$ number of pairs in the same class in C' and assigned to the same cluster in C.
- $b =$ number of pairs in the same class in C', but in different clusters in C.
- $c =$ number of pairs assigned to the same cluster in C, but in different classes in C'.
- $d =$ number of pairs belonging to different classes in C' and assigned to different clusters in C.

Note that $a + d$ corresponds to the number of agreements between C' and C, $b + c$ corresponds to the disagreements, and $M = a + b + c + d = \frac{n(n1)}{2}$ is the total number of unique pairs.

The Rand index results in an evaluation in the range $[0, 1]$ based on the fraction of pairs for which there is an agreement:

$$R(C', C) = \frac{a + d}{a + b + c + d}$$

	2	3	4	5	6	7	8	9	10
NMI measure	0,09	0,17	0,16	**0,19**	0,15	0,17	0,16	0,18	0,17
purity index	0,58	0,62	0,63	**0,66**	0,64	0,65	0,64	**0,66**	**0,66**
rand index	0,46	0,56	0,61	**0,62**	0,61	**0,62**	**0,62**	**0,62**	**0,62**

Table 4.8: External quality measures at different k

Experiments: The results of the experiment are shown in Table 4.8. The NMI measure has the best value at $k = 5$, the purity and the Rand index have the same, best value at different k, but both include $k = 5$, see Table 4.8 the best values at each index are set in bold. So we decide to use $k = 5$ for the experiments.

Finding good representatives Secondly we want to check if a change of the clustering algorithm leads to an increasing of the quality of the clustering. So we have to find a good clustering algorithm for the used dataset. As good quality measure of the clustering we used the same indexes as before. We clustered the dataset with the normal k-means, a spherical k-means and a fuzzy c-means algorithm as partitioning algorithms. We used the Non-negative Matrix Factorisation (NMF) technique that at first minimises the Kullback-Leibler divergence and second minimises squared Euclidean distance (ED). We also tried a kernel k-means clustering algorithm. As result of the three measures, it can be said that the best clustering is produced by the bisecting k-means algorithm, see Table 4.9. The clustering with the bisecting k-means algorithm produces the best values (bold face) at all three used quality measures.

	bisecting KM	KM	spherical KM	fuzzy CM	NMF (K-L)	NMF (ED)	kernel KM
NMI measure	**0,19**	0,14	0,15	0,05	0,04	0,11	0,15
Purity index	**0,66**	0,62	0,64	0,58	0,58	0,61	0,63
Rand index	**0,62**	0,59	0,59	0,41	0,57	0,57	0,59

Table 4.9: External quality measures at different algorithms (k=5)

Now we have found a good representative clustering algorithm for the partitioning clustering algorithm group; the bisecting k-means. For this algorithm we have also found a good number of partitions to this dataset.

In contrast to the nature of the bisecting k-means algorithm, we decide to used the density-based IncrementalDBScan algorithm from [Ester 00].

We have already shown the results with the bisecting k-means algorithm on the dataset 2 and our ”ThemeFinder” in the experiment at Chapter 4.4. That is why we first concentrate only to the other algorithm, the IncrementalDBScan.

4.6.1 Clustering Results with IncrementalDBScan

The IncrementalDBScan algorithm have two input parameters as explained before; the $minPts$ and the eps region value. The $minPts$ defines how many points must be in the defined region around a data point, so that this data point can be declared as "core point". The eps region defines the radius of this region. First we describe our experiments to find out good values for this input parameters for the IncrementalDBScan.

4.6.1.1 Experiments with different eps and $minPts$ values

For clustering the data with the IncrementalDBScan algorithm we made experiments with the dataset from period "1996" and tried to change the values for eps and $minPts$, which give the size of the eps-region and the minimum number of other points in this region, in order to build a new cluster. We have done this tuning experiment to find out a good selection for both input values for the IncrementalDBScan algorithm. We tried short values for both and short values for one of them and a bigger value for the other input value.

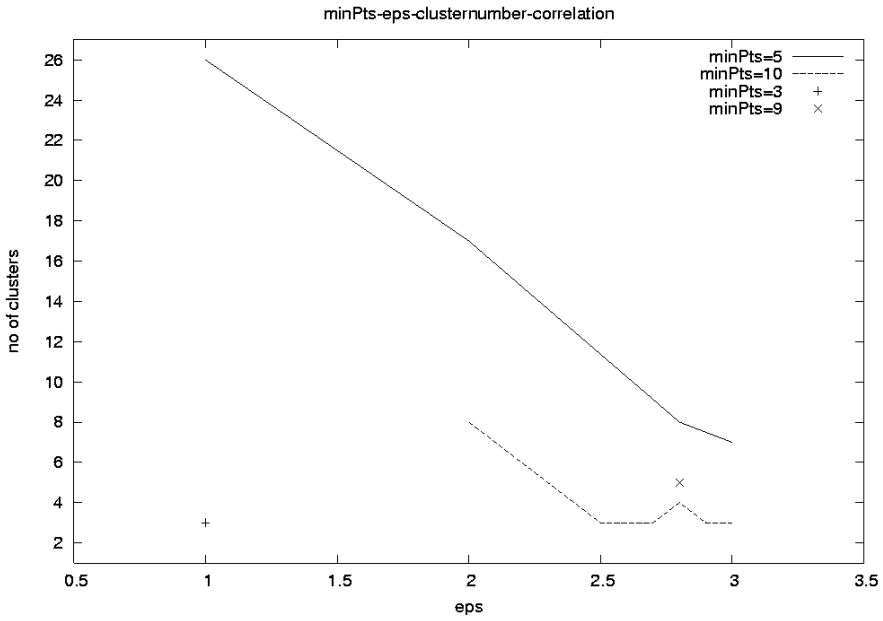
In Figure 4.3(a) we show the results of the cluster numbers at different eps and different $minPts$ where we used the feature space of period 1996 to make a better comparison with the bisecting k-means clustering. The clustering results looks very similar every time. We get a relative big number of noise points, which are not assigned to any cluster and we get different numbers of very small clusters.

On this experiments we get only one small cluster and a big number of noise documents every time. One reason could be the relative small number of documents at period "1996". That is why we made similar experiments with the period "2000" to find out good input values at this period and to find out if the problem with the big number of noise documents is caused by the small number of documents.

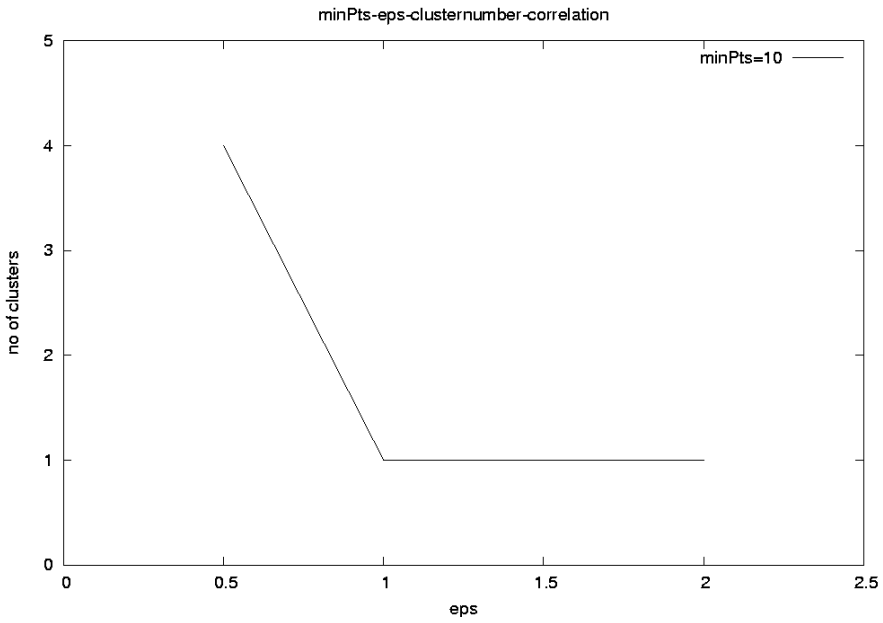
4.6.1.2 Experiments from period "2000"

For the experiments with IncrementalDBScan we decided to start at period "2000". But if we set the period "2000" as start period we have to use the feature space of the period "2000" too as start feature space and so we make short experiments with this feature space and dataset to check the decision and find out a good start value for eps and $minPts$. The results are shown in Figure 4.3(b) for the $minPts = 10$. From this diagram it can be seen clearly that we used $eps = 0.5$ and $minPts = 10$ for the following clustering experiments, so the smallest clusters have a minimum of 11 members. For smaller values at $minPts$ we did not get acceptable results.

We limited the feature space size to the most 100 used words for these experiments. At the cluster results we see that we have many "noise" data points, which are not a member of a cluster. But with increased size of the data set during the time periods the relative size of the "noise" data points decreases. We concentrate only on the clusters found by the algorithm, and on the labels of the clusters. We recognised that we could not produce a label for only one found cluster. After a detailed view to this cluster we



(a) $FS = 1996$



(b) $FS = 2000$

Figure 4.3: Cluster numbers at different eps and minPts

see the reason. The members of this cluster are all documents which are represented by a null-vector by the used feature space. The null-vector can be created at the vectorisation step because we use only the top 100 words of the dataset for vectorisation. Documents which not consist of one word of this top 100 words are represented by null-vectors. So it was acceptable that this cluster has no label.

The following Table 4.10 shows the number of clusters of each period and the number of found again clusters of the previous period via ”ThemeFinder”.

period	2000	2001	2002	2003	2004
no.of clusters	3	6	12	27	41
matched clusters		3	6	12	26

Table 4.10: Results of matched clusters with the IncrementalDBScan

It can clearly be seen that we have no matched clusters at period “2000” with a previous period because “2000” is the start period. We found all clusters of the previous period again in the next period with the exception at period “2004” where we found all of the 27 clusters of period “2003” except of one. Another effect of rediscovering nearly all clusters at the next period is that the labelled clusters at one period are persistent over future periods, for example the 3 labels of period 2000 are also present on period 2004. It could be mentioned that in every new period new clusters showed up because at earlier periods there were not enough documents about the topic of the new clusters and the documents are noise points at early time periods. So the relative size of the noise decreases.

These results that nearly all cluster labels are persistent over the following periods and new themes arise only through new cluster labels, confirming our assumption, which we had before we started with this algorithm, caused by the special type of incremental density based clustering algorithm.

4.6.2 Comparison of the ”ThemeFinder” with two different clustering algorithms

Because the used clustering algorithms are extremely different from their nature, we compare the results and so the influence of the clustering algorithm used for the monitoring process with our ”ThemeFinder”. Both algorithms found labels with our method for label creation and label monitoring with the ”ThemeFinder”. A comparison of the labels is not easy because the number of labels at the bisecting k-means clustering results is static and with the IncrementalDBScan it is variable. This leads automatically to relative stable labels with the IncrementalDBScan and to labels with lower stability at bisecting k-means algorithm results.

We have 4 labels at a maximum at the bisecting k-means results and found 2 of the labels at the periods “2000” and “2001” again at the labels at the clustering results from the IncrementalDBScan algorithm. For the periods from “2002” and later we found 3 labels actually at both label sets.

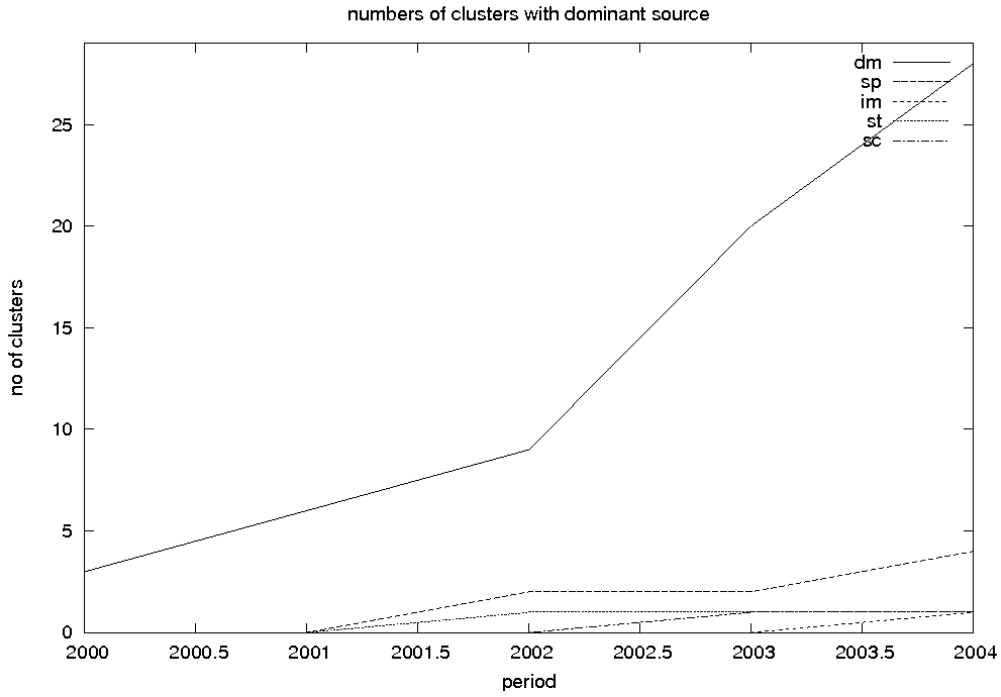


Figure 4.4: Clusters with dominant source subgroup

Secondly both cluster algorithms produce the dominant clusters from the dominant group. The number of documents from the “data mining” subgroup becomes more and more dominant with time. If we evaluate the cluster labels to the subgroups by calculate the percentage of documents in the cluster from which subgroup they originate, we found 2 labelled clusters from the “data mining” subgroup with the bisecting k-means. With the IncrementalDBScan algorithm all clusters at period “2000” and “2001” originate from the subgroup “data mining” and in the following periods the percentage of labelled clusters from this group is very high.

In Figure 4.4 we show the number of clusters where the most documents ($> 51\%$) come from one subgroup. The shortcuts at the lines stands for the names of the subgroups, data mining (dm), spatial databases (sp), image databases (im), statistical databases (st) and scientific databases (sc). The differences between the number of clusters at each period from this figure and Table 4.10 arise because we have some clusters which have a mixed composition from the several subgroups and here we list only the clusters where the majority originate from one group.

We also see that changes at the labels are found by monitoring the labels with the ”ThemeFinder”. But the differences between both clustering algorithms are in the kind of changes which we found. At the clustering with the bisecting k-means the label of a cluster changes during the time, because words as member of the label become non-members and other words become new members of the label. In contrast to this, with the IncrementalDBScan, the labels of a cluster are very stable but the number of clusters increases and so from one to another period new labels will be created and show new

topics.

Result of the comparison We present the results of the ”ThemeFinder”, ones with the bisecting k-means and ones with the IncrementalDBScan clustering algorithm. As a main result, we see that ”ThemeFinder” is useful at both algorithms to detect and monitor the topics of the clustering. The big size of noise data as one result of the IncrementalDBScan shows us, that the used data are very noisy and so it could be one reason for rather negative clustering results with the bisecting k-means algorithm. This is already known as one disadvantage of this algorithm.

Another result is, that we can see the change of topics in the data, when monitoring the labels at both algorithm results. The difference is, at the bisecting k-means we see it on the change at the labels itself and at the IncrementalDBScan we see it on the labels of the new created clusters from one to next periods.

We have also seen that the number of adjusting the feature space is very small. At bisecting k-means it was over 9 periods necessary only two times and over 5 periods with the IncrementalDBScan zero times to adjust the feature space.

All results of this comparison leads us to the statement that the ”ThemeFinder” can work with different clustering algorithms.The selected clustering algorithm has an influence to the kind of results depending on the characteristics of the algorithm. This characteristics should be noticed at the interpretation of the monitoring results with the ”ThemeFinder”.

4.7 ”ThemeFinder” with different labelling methods

The ”ThemeFinder” uses only the labels of a clustering as input data. So basically the clustering algorithms and the labelling methods are not predefined by our framework. At the experiment 4.6 we have shown that clustering algorithms, which are very different from their nature, can also be used with our framework to monitor labels over time with the ”ThemeFinder”.

At this experiment we will study the influence of the labelling methods used on the monitoring process and the quality of the monitoring results with our ”ThemeFinder”. To study the influence of different labelling methods, we use the labelling methods, which are explained in Chapter 2.2.2, these are the TOP-, the IGAIN and the χ^2 labelling methods.

For the experiments, first we have to select one clustering algorithm, whose result is the input for all three labelling methods. After usage of the same quality measure from the experiment above, which were also implemented in TCT, we decide to use the *Non-negative Matrix Factorisation* (NMF) with the *Kullback-Leibler* divergence, also referred too as relative entropy [Lee 99]. The decision for this algorithm is caused by technical reasons although we know that this algorithm do not produced the best results at the experiments before. All used labelling methods from the used TCT tool can used the results of this algorithm to create labels for the clusters. But the objective here is

the influence of the labelling methods to the results and not the quality of the clustering algorithm itself. As the distance measure the Euclidiance distance is employed.

A comparison of the different cluster algorithms at TCT can be found in Table 4.9. There we compare the cluster algorithms with different internal and external quality measures. The NMF algorithm has shown good quality values at this comparison. One other reason for the decision to use this algorithm is that this is the best of a small number of clustering algorithms for which all three labelling methods are working at the TCT clustering toolkit. For example the bisecting k-means does not work with all three labelling methods at this toolkit. The reason is the internal structure of the clustering results at the TCT tool.

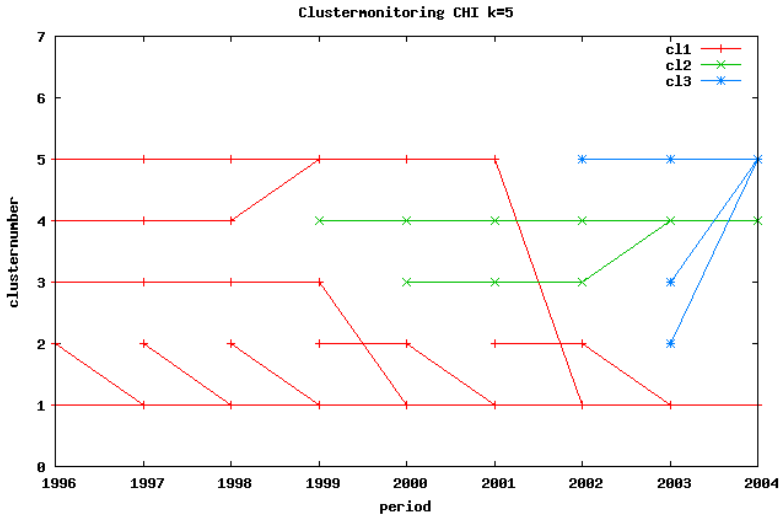
The next step is to define a good value for the cluster number. So we make the quality test with different values for the cluster number k. We created clusterings for $k = \{2, ..., 10\}$ and found out that the clusterings with $k = 5$ and $k = 6$ are clearly the best clusterings from the view of the different quality measures. So we make the experiments with both cluster numbers $k = \{5, 6\}$. As result of the labelling methods and the usage of "ThemeFinder" at the clustering results with $k = 5$ and $k = 6$ we recognised that we get much better results for $k = 5$, which is also the number of different subcategories at the ACM hierarchy, and so we present in the following the results for each labelling method at cluster results with $k = 5$. This is the same value which was suggested from the results of the previous experiments with the bisecting k-means algorithm.

Because one of the goals of the "ThemeFinder" is to minimise the cost expenses for new generation of a feature space, first we use the "ThemeFinder" without the feature space adjustment feature. It means that at every time period the actual feature space of this period is used. The assumption behind that is that we will study the influence of the labelling method itself and so we guarantee that at every period we use the best possible clustering to create labels. At the second experiment we use the "ThemeFinder" on the clustering results with the feature space adjustment functionality. So the feature space of a period is only created if the good feature space of the previous period is not good enough compared to our quality definitions. This second experiment is the comparison of the influence under the supposed usage conditions with feature space adjustment only if it is needed by the quality restrictions.

4.7.1 Results without feature space adjustment

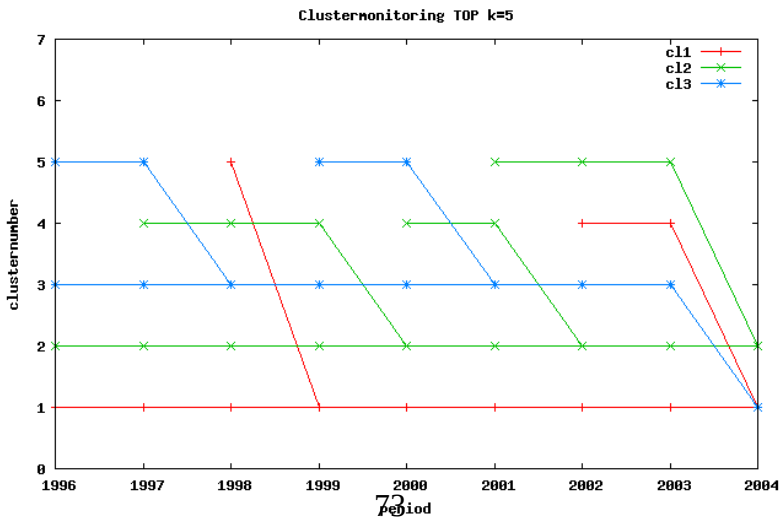
At first we vectorised the documents of all time periods of the dataset 2 with the actual feature space of each time period. After this we cluster the document of each time period with the NMF clustering algorithm and create labels with the different labelling methods. After this we start our "ThemeFinder" to find the corresponding labels from one period to the next period, starting with period 1996. The results of "ThemeFinder" are presented in Figure 4.5.

We see that the "ThemeFinder" has found three topics (topic 1, 2, 3) with the "TOP", three topics (topic 1, 3, 5) with the "IGAIN" and two topics (topic 1, 5) with the χ^2 labelling method labelled clustering results. These topics are relative stable over more



(a) χ^2

(b) IGAIN



(c) TOP

Figure 4.5: Results of ”ThemeFinder” without feature space adjustment

periods. Most of them are not persistent themes according Def. 5 but themes according Def. 6. It can be seen that for the ”TOP” method three topics exist over the whole monitoring period, excluding period 2004, and at both other labelling methods one or two topics arised during the monitoring process, which can be called as persistent themes. All the method results have in common that many cluster labels of a period are merged together in one of the feature periods. For example in every figure of Figure 4.5 the lines for cluster label one and which cluster labels merge into it.

Intuitively it is clear that not all clusters over all periods have persistent labels, because we expect a change of topics over time at the document archive. We think, the result with three persistent, at least nearly every time persistent, labels is a really good result for these experiments, especially if we compare those results with results from experiments described in Chapter 4.2 and Chapter 4.4 ([Schult 06b] and [Schult 06a]). There we have used other clustering algorithms and the results of ”ThemeFinder” at the same dataset, the dataset 2, has reached same quality at best. Because we did not use the feature space adjustment property of the ”ThemeFinder”. At every time period we used the best possible feature space for creating the document vectors as document representation. So it is obvious clear that the clustering results and following that also the monitoring results have the same quality at best if we use the feature space adjustment described in our ”ThemeFinder”.

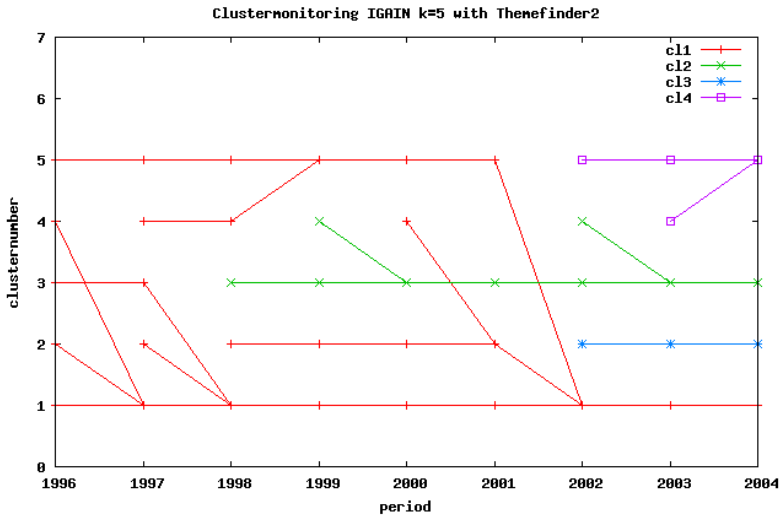
4.7.2 Results with feature space adjustment

The first result of the usage of ”ThemeFinder” with feature space adjustment and different labelling methods is the label creation by the χ^2 labelling method returning unusable data. With this labelling method we get the same label for every cluster and the statistics of the words of the labels are always 0.0. We did not find the reason for this problem; if it is a problem of the used tool, an implementation problem or a problem of the used data. So we can not use the χ^2 labelling method for the ”ThemeFinder” with the time and cost-saving method of feature space adjustment. This problem with the χ^2 labelling method is not really shortcoming for our ”ThemeFinder”, because the created labels are the input to this algorithm. But at this step we cannot create any useful labels, so the user has to employ another labelling method.

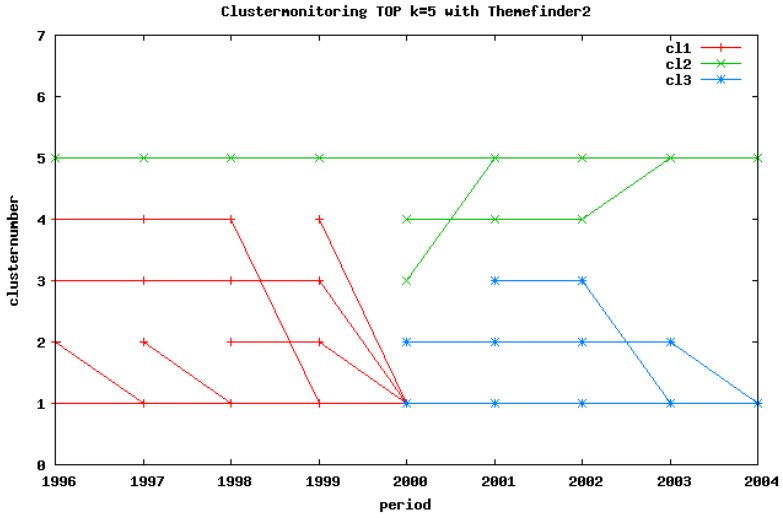
Both other algorithms are usable and the results are presented at Figure 4.6.

The ”IGAIN” labelling method results are shown in part (a) of this figure. These results are produced by adjusting the feature space only four times at the monitored 8 periods. So by these results it save nearly 50% of the cost for producing an actual feature space. In the Figure 4.6(a) it is clearly visible, that one permanent topic exists, to which at many periods other clusters are merged. As a next point we see that the topic of cluster 3 is also persistent since period 1998. In comparison to the results of usage ”IGAIN” without the feature space adjustment, we see that we get different but similar results and we find also the main dominant topics.

Concerning results with the ”TOP” labelling method, presented in Figure 4.6(b), we see one persistent topic over the time (cluster 5). The adjustment of the feature space has to be done also only four times, the same number as with labelling with the ”IGAIN”



(a) IGAIN



(b) TOP

Figure 4.6: Results of ”ThemeFinder” with feature space adjustment

method. At the first time periods until period 2000 we see one more persistent topic (cluster 1) which has the same properties as the cluster one of the results without the feature space adjustment. The point is interesting, that in so far as from period 2001 we visit a new cluster 1 with the same properties as the cluster one of the first periods, but between the cluster one of period 2000 and the cluster one of period 2001 no connection was found by ”ThemeFinder”. This allow us to suggest that between these both clusters the connection is lost through the usage of the feature space adjustment.

4.7.3 Comparison of the results with different labelling methods

As comparison of the three labelling methods we have to make a difference between using the feature space adjustment or if not.

If we do not use it, we can see, that ”ThemeFinder” produce similar results with all three algorithms. We can monitor the labels, produced by all three methods, and can find topics which are present at more periods and can also find changes in existing topics. If we should rank the labelling methods we would prefer ”TOP”, behind is the ”IGAIN” labelling method and at last the χ^2 method. But the differences are not really significant.

If we use the feature space adjustment option with ”ThemeFinder”, we see that the χ^2 method is not able to produce useful labels for the clustering results and so this method is unusable for topic monitoring with our ”ThemeFinder”. The ”IGAIN” and the ”TOP” labelling method produce labels which the ”ThemeFinder” could use to create useful results and reach nearly the same quality as without the feature space adjustment option with both labelling methods. But both need a new creation of the feature space only at 50% of the time periods and so they are not so expensive as the experiment without feature space adjustment and those labelling methods. From the point of the quality of the results we see that the ”IGAIN” methods results are a little bit better as the results of the ”TOP” method.

4.8 Comparing themes to evolving clusters

The goal of our experiments is the observation of detected topics, how the topics change over time. The experiments before have shown that we can detect and observe topics with our algorithm ”ThemeFinder”. At the experiments in Chapter 4.2 and 4.4 we make an empirical evaluation of our labels found against the original category names from the ACM hierarchy as ground truth.

Now at this experiment we compare the results of the experiment from Chapter 4.4 with the results of a modified version of the FOCUS framework from [Ganti 99b], which is a good framework for observing clusters over the time, if you take a detailed look on the cluster members. We make this comparison because we assume that monitoring of clusters should be produce similar results as the monitoring of the labels which are produced from the clusters. So this comparison can be seen as another kind of evaluation of the monitoring results with the ”ThemeFinder”. With our ”ThemeFinder” we save time

and complexity to monitor clusters over time. Especially at big datasets the complexity of FOCUS is much higher as that of "ThemeFinder". If we have labels already it should be clear that the comparison of different labels is much faster as a comparison of clusters of many documents because the calculation of the union at two clusters is much more expensive as the calculation of an union of two labels. A label consists only of some words and a document cluster has normally many more documents as cluster members.

4.8.1 Adapted FOCUS framework

For comparison of our results with "ThemeFinder" about the label monitoring over time with FOCUS, we adapt the FOCUS framework from [Ganti 99b]. The FOCUS framework is a well known framework to detect similar clusters at a collection of clusters to a given cluster. The goal of this experiment is a comparison of the quality of our monitoring process. On the one hand it could be that we do not monitor all clusters over time that are possible to monitor. On the other hand, it could be that we monitor clusters which are similar by their labels but their content has no similarities and so they should not be monitored together.

The results of the FOCUS framework should show us, that we observe the correct labels of the same clusters over the time and not that we observe label of different clusters, which look like the same at different periods and that we monitor all monitorable clusters with our framework too.

We calculate the percentage of how many documents of a cluster C_i exist in C_{i+1} according to the following, equation(4.1) and call it $foc(C_i, C_{i+1})$

$$\forall(C_i \in \zeta_i, C_{i+1} \in \zeta_{i+1}) : foc(C_i, C_{i+1}) = \frac{|C_i \cap C_{i+1}|}{|C_i|} \geq \tau_{foc} \tag{4.1}$$

The threshold τ_{foc} adjusts, how many percentage of the cluster C_i must be present in the new cluster C_{i+1}, so we say that the cluster C_i is also present in the new period t_{i+1} and has survived. We selected the cluster-pairs (C_i, C_{i+1}) with $max(foc()) \wedge foc() \geq \tau_{foc}$ and say that C_i has evolved to C_{i+1}. If $\tau_{foc} \geq 0.5$ at a maximum only one new cluster exists, in which the cluster C_i has evolved. Otherwise it could be possible that more than one cluster at period t_{i+1} has the same value of $foc()$ to a given cluster from period t, which is comparable with a split of the cluster.

We used the results of this adapted FOCUS framework to compare with our results. The FOCUS framework is algorithm independent and it uses the clustering results only for the monitoring, like our "ThemeFinder" which uses only the set of labels of an existing clustering independently from the used clustering algorithm as we have shown before the special results with the different clustering algorithms at sub-chapter 4.6.

4.8.2 Comparison with FOCUS

We used the adapted FOCUS framework and our "ThemeFinder" on the same clustering results. The data preprocessing, the vectorisation and the clustering step are the same

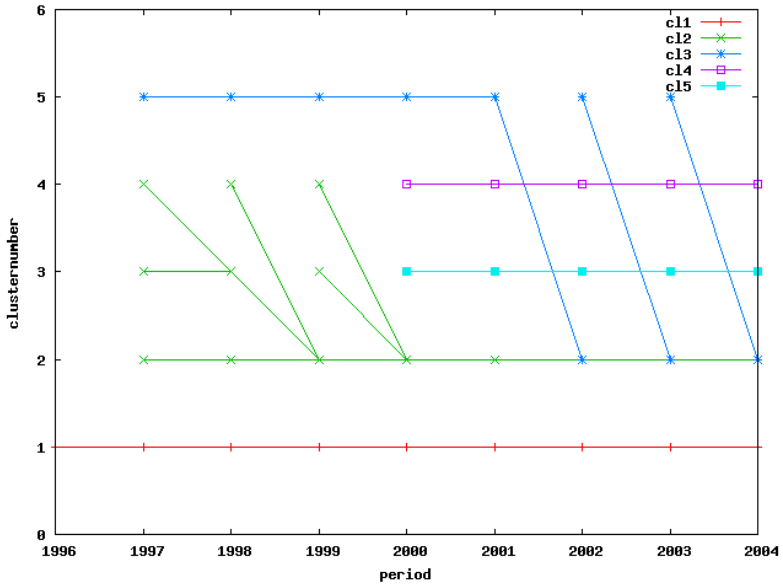
for both frameworks. For the adapted FOCUS framework we use these clusterings to monitor the clusters over time. The result of this monitoring process is shown in Figure 4.7(a). At the same clusterings we start the labelling process and then our framework with the ”ThemeFinder” to monitor the labels over time. This results are shown in Figure 4.7(b). At the adapted FOCUS results we can see that some clusters of one period merged together into one cluster at the next period. One example is the clusters 3 and 4 at period 1997 which are merged at period 1998 into cluster 3, which also merges with cluster 2 at period 1998 into cluster 2 at period 1999. Another interesting point is, that after a merge new clusters exist and sometimes they also are merged later. The new clusters are raised by the clustering algorithm. The bisecting k-means, with a fixed number of clusters, produce a new cluster at the new time period if at the period before two cluster merged.

The lower diagram of Figure 4.7 shows the results of our ”ThemeFinder” after the label matching described in Chapter. 3 on the dataset 2 and the cluster results of the experiment before described in Chapter 4.4. As we can see, the ”ThemeFinder” follows nearly the some clusters as the adapted FOCUS framework. There are two basic differences between the usage of the adapted FOCUS framework and the framework with the ”ThemeFinder”.

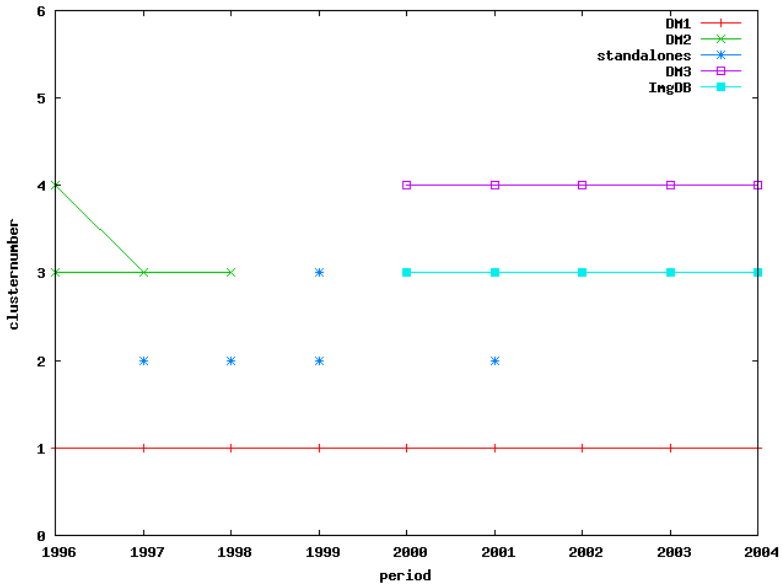
- only the labelled clusters and not all clusters of the clustering are present when the cluster labels are monitored with ”ThemeFinder”. So we see in the (b) diagram a smaller number of clusters.
- normally we do not detect splits of a cluster, because our *best_match* algorithm only finds one cluster as best match to another cluster.

Caused by the nature of the *best_match* algorithm we cannot find splits because only the best corresponding label of the next period to a given label will be found and not the second best too, which is needed to define a split of a label. This supposes that we use the *best_match* algorithm forward at time and find the best matching labels at time period $t + 1$ to the given labels of period t. If we use this algorithm backward at a time, we can not find merges but splits, because one cluster of period t can be the best matched cluster for two clusters of period $t + 1$. If the *best_match* algorithm has as result the same cluster at t_{i+1} for two different clusters at t_i, then we can say that both clusters at t_i merge together to the best match cluster at t_{i+1}. In the diagram we see such points at cluster 3 and 4 at period 1996, which merge to cluster 3 at period 1997. We see that the label “discovery knowledge datum” at period 1996 change over time to the label “datum mining” at period 2001 and “datum” at the following periods. Here we see that our assumption is correct that the terminology of this document archive changes over this long time.

Evaluation of the goal Comparing the results of the adapted FOCUS framework with the results of our framework and the ”ThemeFinder”, we see that our framework monitors only the labels of clusters, which are also following clusters by the FOCUS results, for example, the cluster number 1 at both diagrams in Figure 4.7 or the clusters



(a) adapted FOCUS



(b) ”ThemeFinder”

Figure 4.7: comparison of FOCUS and ”ThemeFinder”

3 and 4 from period 2000 until 2004 in both diagrams. It is clear that not all clusters, which are monitored by FOCUS can also be monitored by our framework. One reason could be that we have not for all clusters an useful label with our framework and so these clusters cannot be monitored by our framework. But we see that nearly all monitored clusters by the FOCUS also monitored by our framework too.

As result of this experiment we can say that our ”ThemeFinder” achieves good results similar to the FOCUS framework from Ganti et al [Ganti 99b]. But with our algorithm the user does not need to take a detailed look into the real data. The user only has to take a look to our observation of the labels. An other advantage of our ”ThemeFinder” is the minimisation of the adaption of the feature space. The reason that it is a advantage we explained already earlier in this Chapter. Another advantage our ”ThemeFinder” is the possibility to handle dataset, which have no overlapping at the different time periods, like the dataset 1 at our experiments. Also at this datasets ”ThemeFinder” can monitor the cluster labels. FOCUS has no possibility at such a dataset, because at FOCUS an union of two clusters of different time points is calculated.

4.9 Coverage of the ACM section H2.8.

As last part of this experiment chapter we will make a comment to the results with the selected dataset, the ACM sub-archive from section H.2.8. It can be seen in Table 4.7 that the coverage with regards to the ACM sub-archive is rather low and that some clusters reflect subtopics rather than whole categories.

We first checked whether the low coverage can be attributed to the clustering algorithm. As already mentioned, we have experimented with different clustering algorithms. In [Spiliopoulou 06], we have also considered $k = 10$ and discovered that this larger value finds more informative subtopics (obviously) but still does not allow for the identification of all classes. Hence, we performed a series of classification experiments, i.e. used the document labels, and searched for features/keywords with high predictive power.

Similarly to many clustering algorithms, classification algorithms like C4.5 and Naive Bayes require special tuning to deal with highly skewed data. Therefore, we have first attempted a separation of the dominant class “Data Mining” from the rest of the collection and then tried to build a classifier for the remaining classes. We concentrated on data from one period (2001). The separation of the “Data Mining” class from the others was achieved with an accuracy of more than 80%. This reflects that the identification of this class in the data is easy - a fact that is apparent in our clustering results as well. However, the classification accuracy for the other four categories was low. One of the most remarkable results was that the SVM and the J4.8 classifiers (the WEKA implementation of the C4.5) assigned the documents of the category “scientific databases” to the class “spatial databases”, while Naive Bayes assigned a large portion of documents on spatial, statistical and image databases to the class “scientific databases”. Hence, we came to the conclusion that the categories cannot be properly separated, most likely because of the existence of subcategories. The subtopics found by ”ThemeFinder” (association rules,

image retrieval) are indicators of such subcategories.

Summary In this chapter we show at experiments that our ”ThemeFinder” works well with datasets, which have different forgetting strategies. But also we show that the ”ThemeFinder” can be used with different clustering and labelling methods. Only labels must be created for the clusters, then the ”ThemeFinder” can be used for monitor the cluster labels over time. As last experiment we compared the quality of the monitoring results of the well known FOCUS framework with our monitoring results and pointed out that our framework reach similar quality results but have different advantages compared to the usage of FOCUS.

5 TheMoT - the Theme Monitoring Tool

At Chapter 3 we have described our complete framework including the algorithm ”ThemeFinder”. At experiments in Chapter 4 we showed that our framework works well. As we mentioned at the experiments, it exists already some tools for many steps of our framework, like for clustering or labelling. That is why we concentrate on the developing of a prototype for the main part of the ”ThemeFinder”, the method ”best_match()” to see as result a graph of the cluster evalution.

In this chapter, we present the Theme Monitoring Tool ”TheMoT”. This tool is designed to make the label matching process very easy and show the results of this matching process. TheMoT consists of two parts. The first part is the ”$best\_match()$ - Tool to find the corresponding labels from one time period to the next. The second part is the presentation of the total monitoring process over all periods.

First we introduce the usage of the graphical user interface of the ”best_match”-Tool before we show the visualisation tool used for monitoring the label evolution over all periods. At end we give some technical details for the input parameters of both tools.

5.1 Matching for the Monitoring

The user of our framework has a list of labels for each time period as the result of the clustering and labelling process. For finding the matching labels from one time period to another, the user needs the file with the list of labels for each period. This label list files are the input to the ”best_match”-Tool .

The ”best_match”-Tool has two different interfaces which can be invoked by the user of the application. As one interface we have implemented a command line interface. This has the advantage that the user can integrate the algorithm itself into other tools. Otherwise the command line allows an easy and fast access to use the ”best_match”-Tool. The command line interface is called with the two label list files as input parameters.

As other interface we have developed a graphical user interface (GUI) to use the algorithm ”best_match()”. This has been developed for a more detailed view to the mapping results.

Now we will briefly introduce the GUI for ”best_match”-Tool only, because the command line interface is only the call of the program with the both label list files for the selected time periods as input.

5.1.1 The GUI of ”best_match”-Tool

To start the ”best_match”-Tool the user has to execute the file *GuiMain.py*. This can be done using the linux operating system by typing *python GuiMain.py* or on a windows system by double clicking on the *GuiMain.py* file. At start of ”best_match”-Tool a window like in Figure 5.1 is presented to the user.

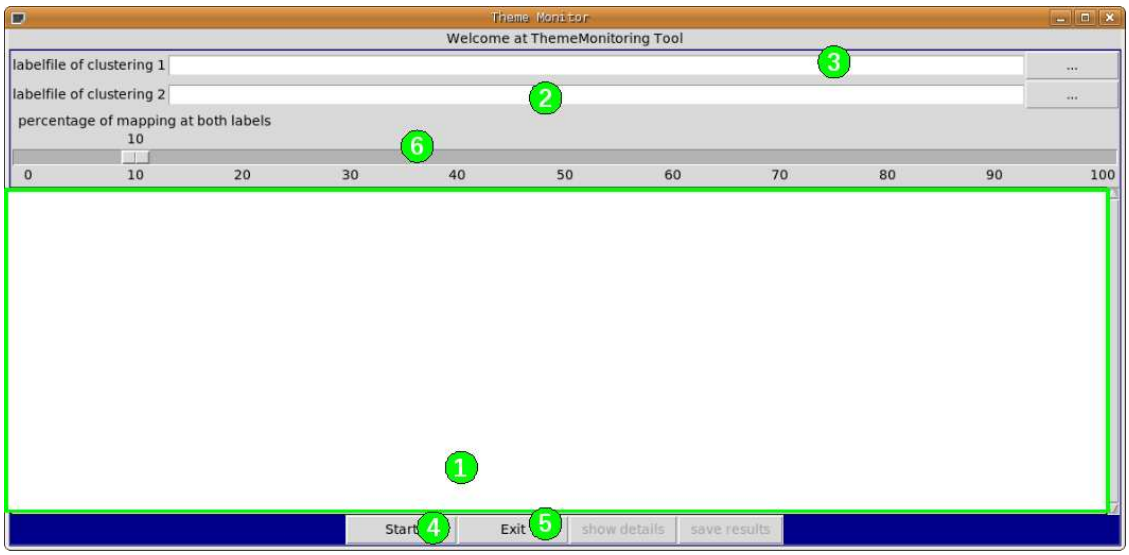


Figure 5.1: ”best_match”-Tool at start

At this start window the user sees on the top two input fields and beside of each a button to open a file selection box to select the both label list files of the clusterings, which the user will analyse. The first input field is for the label list file of period 1 (green 3) and the second for the label list file of period 2 (green 2). Below these two input fields, a slider is presented to adjust the threshold $\tau_{deviation}$ (green 6), which specifies how many percentage of the compared label must be equal to say this is a match. The second half of the starting window is a big empty text box under the slider (green box 1). This box is the result box used to present the results of the $best\_match()$ algorithm. At the bottom of the main window the user sees four buttons, the ”start” button to start the algorithm (green 4), the ”exit” button, to exit the program (green 5), the ”show/hide-details” button to show/hide detailed results and the ”save results” button to save the results to an external file, the matching file. The last both buttons are not active at program start. They become active after the algorithm was started.

To start the ”ThemeFinder”, we select both labelling files of the different clusterings, we will monitor. We can do this by typing the path to the files into the input fields or selecting the files via the file selection box by clicking the buttons beside the input fields. Then we select the percentage of the mapping of the labels via the slider in the middle of the window and press the start button at bottom. After starting the *”best_match()”* algorithm the result of the monitoring process will be shown in the result box. Here we see first the short result list as an overview of the matching results (see Figure 5.2). The

short result list shows only the numbers of the labels at the selected label list file and not the label itself. The label numbers of the first label list file grouped by the label will be shown which was recovered at the second label list file. At the result box this is shown in the ”matched found:” section (green 3). Also we see which labels of the first label list file are not found again, shown in the section ”died” at the result window (green 4) and which labels of the second label list file are not a mapping part to the first label list and so we group these labels under ”born” at the result window (green 5).

In Figure 5.2 we have selected a file with the name ”k5-1996.txt” as the first label list file. The file ”k5-1999.txt is selected as second label list file. In this figure we see that the label number 1 matched to label number 1, label 2 matched to label 4 and label 3 and 4 matched to label 5. The last matching is an example for a merge of two labels. The labels 3 and 4 of period one merge to label 5 at period 2. The label 5 from period 1 has no matching label in period 2 and so this is shown in section ”died”. The labels 3 and 4 of the second label set file are no matching labels and so we group them to section ”born” labels. This short result list can be saved to an external file by pressing the ”save results” button (green 2).

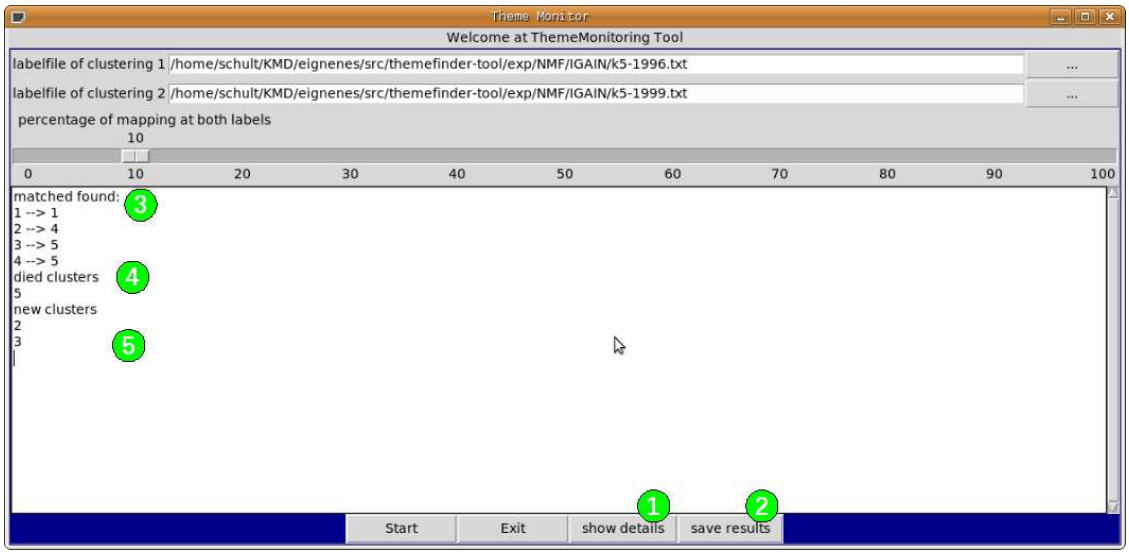


Figure 5.2: ”best_match”-Tool with short results list

We can switch between the short result list to the detailed result list by pressing the button ”show details” (green 1). The details of the results are shown in Figure 5.3. We show the labels itself and not only the label numbers at the result box (green 2). This is the main difference between the short and the detailed result list. The main content of the result box, the three sections (matched, died, born), are equal at the short result list and the detailed list.

In the ”matched found” section, the words of the labels, which are equal in both labels, are coloured green, the other words are red (green 2). So we can quickly detect the matching parts of the labels. The button ”show details” has changed to a button ”hide details”, which switches back from the details result list to the short result list.

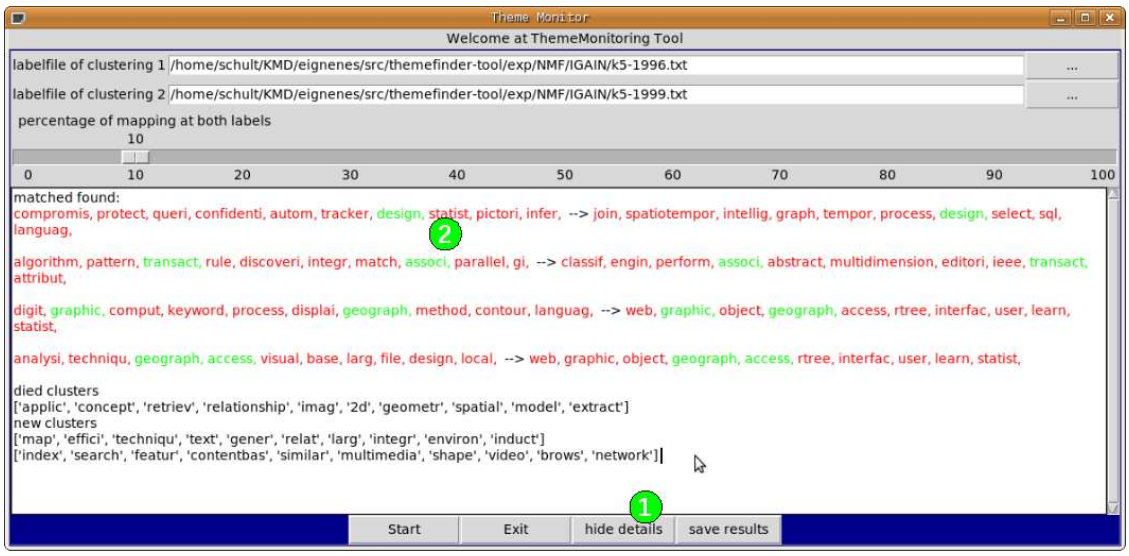


Figure 5.3: ”best_match”-Tool with details results list

With the ”best_match”-Tool the user of the label monitoring process can monitor the label changes from one time period to the next. To monitor more than two time periods we have developed a visualisation tool which we describe in the following sub-section.

5.2 Visualisation of the Monitoring

The ”best_match”-Tool looks for the matching labels from one time period to the next. Normally in such applications we do not have only two time periods, we have many more (in our experiments we have at maximum 10 time periods). To visualise the evaluation of the labels over all time periods, we developed a visualisation tool for our TheMoT.

The visualisation tool consists of tree inutitive steps:

Step 1: Selection of the matching files from the ”best_match”-Tool and timely ordering of this

Step 2: evaluation and calculation on the data

Step 3: presentation and export of the graph

To start the program we execute the program jar file by double clicking on it on Windows-based machines or type *java -jar visual.jar* at the console in linux-based systems. The start screen is shown in Figure 5.4.

On top of the window the three main steps of the program can be seen, as mentioned before, shown with numbers on a red circle. At first we see the selection window (red 1). This consists of three parts; the file browser at the left side (green 1), the options buttons at middle (green 2) and the matching file box at right (green 3). We can browse over the file system of the computer via the file browser on the left side, in order to search for

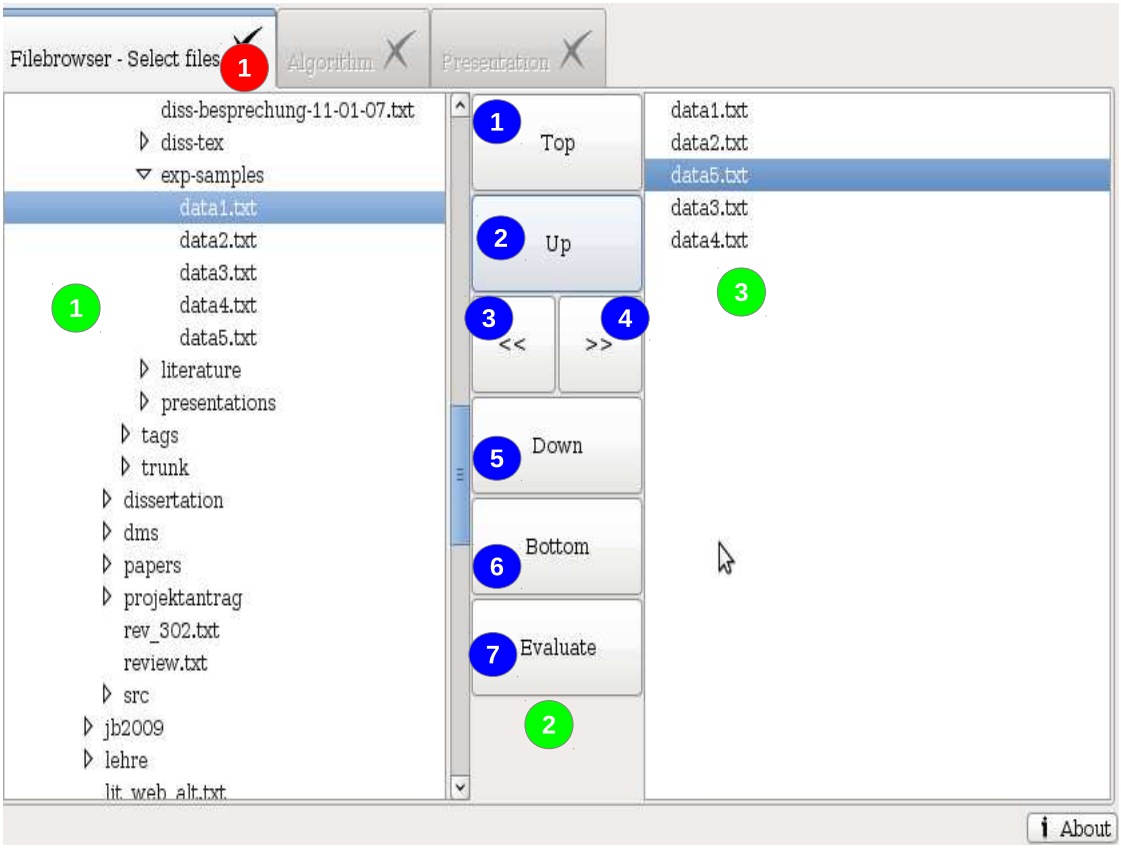


Figure 5.4: start window, step 1

matching files. If we have one matching file selected on the file browser, we can add it to the matching file box (green 3) by clicking the button ”>>” (blue 4) from the options buttons in the middle. A deselection of a file at the matching file box can be done by clicking the button ”<<” (blue 3) after selecting the file. An ordering according to time of the matching files at the matching file box can be done via the buttons ”top” (blue 1), ”Up” (blue 2), ”Down” (blue 5) and ”Bottom” (blue 6). If all matching files are selected and ordered correctly, we can start the analysis process by clicking the button ”Evaluate” (blue 7). All matching files at the matching file box are given to the evaluation algorithm and the main window switches to the second main step, ”evaluation and calculation on the data”, seen at Figure 5.5.

At the evaluation window we see two main components. One is the button ”evaluate collected files” (blue 2) to start the evaluation process. The second is the status window (green 1) where we can see different logging information during the evaluation process. This can be helpful to spot errors during this process. If we activate the checkbox (blue 1), the evaluation algorithm ignores lines with an error at a matching file but the evaluation process is not stopped.

If the evaluation process is successful, the presentation step is available to present the result as a graph, see Figure 5.6. Here we see the main window with the graph as a result

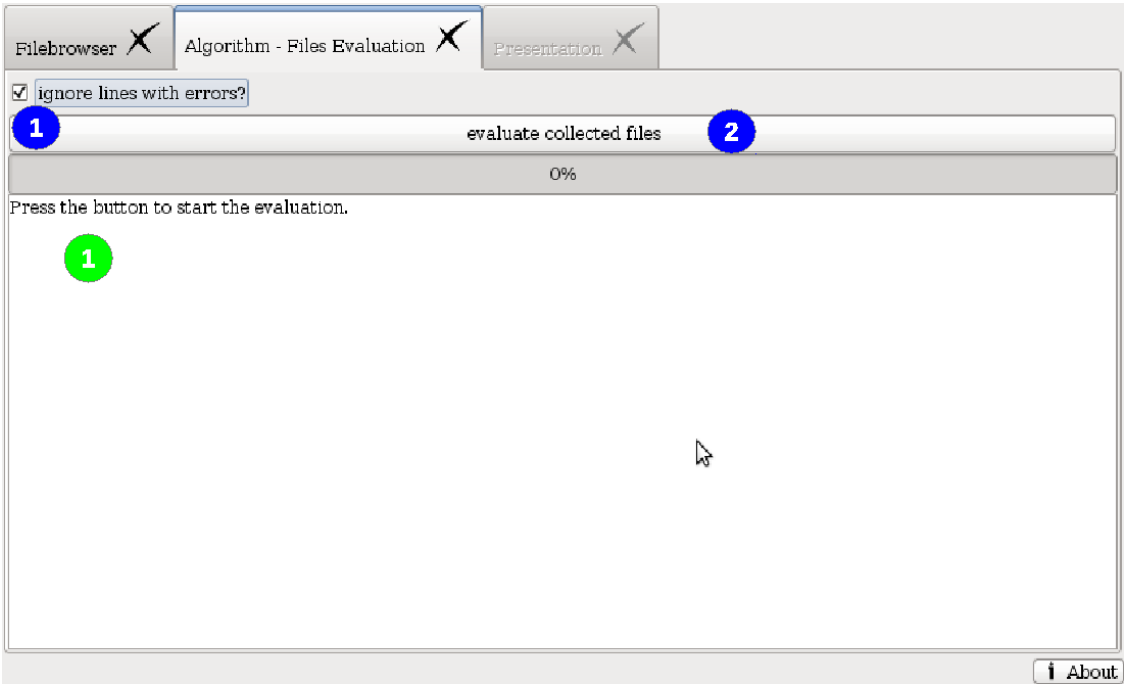


Figure 5.5: Evaluation and calculation step window

of the evalutaion process (green 1). At bottom we see a button (blue 1) to export the graph as a picture in the *jpeg* file format.

We can see at the graph, that 11 different labels were recognised over the complete observed time span, in this example 5 time periods. With the resulting graph it is very easy to see the monitoring results. At this example we see that the labels 3 and 5 from the starting period $t = 0$ are present over the whole time span. We can see that the numbers of the corresponding clusters change over time. The reason for this has its origin in the clustering of the documents at each time period. Most clustering algorithms, which work not incremental, randomly select the cluster-id. At the ”best_match()” method we rediscover again clusters over different time periods, which do not have the same cluster-id at the different time periods.

5.3 Technical Details

In the following we shortly describe some important technical details of TheMoT.

Both sub-parts of our TheMoT , the ”best_match”-Tool and the visualisation tool, are developed plattform independant. The ”best_match”-Tool is written in the programming language Python (www.python.org) because it is very easy and fast to implement a prototype with this language. The visualisation tool is written in *Java* (www.java.com). The main reason for switching the programming language was the easy-to-use java interface for a graph library for drawing graphs. The development was done at a Intel machine with 1 GHz and only 1GB RAM using Ubuntu linux (www.ubunto.com). This shows

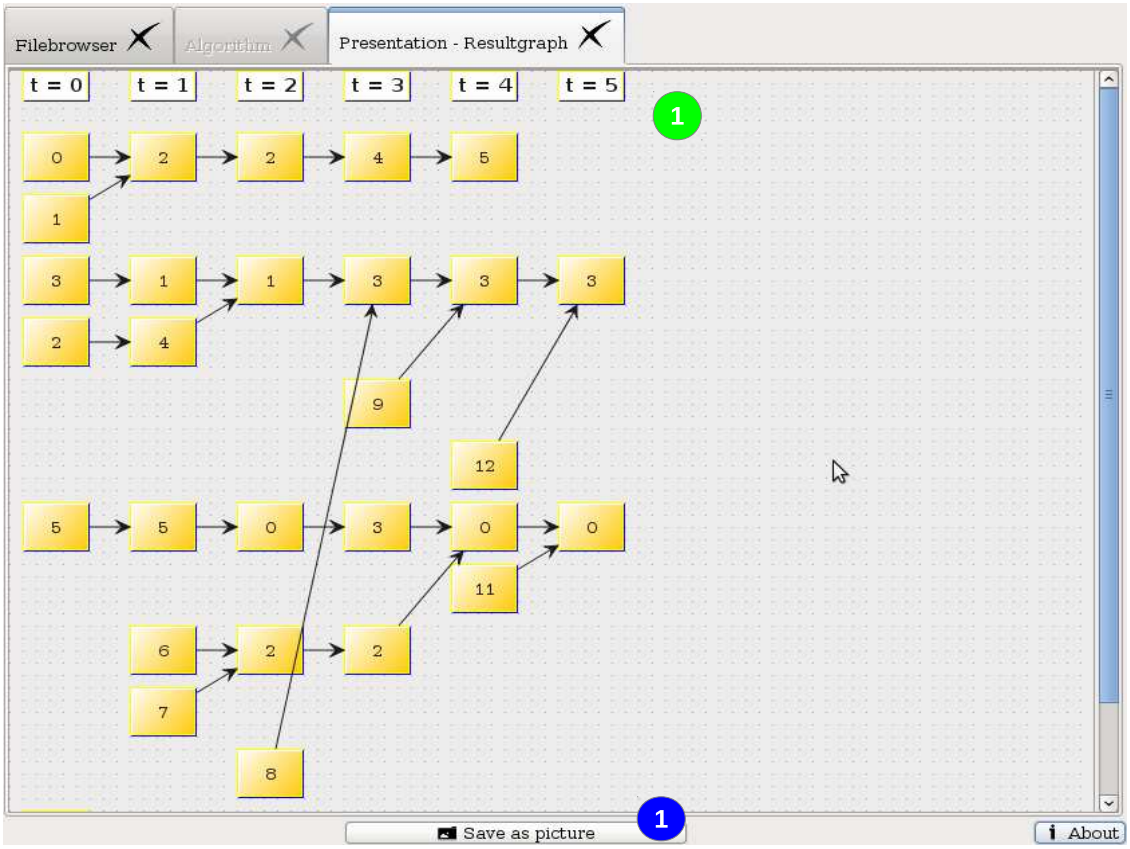


Figure 5.6: the result graph

that both programs do not need many resources.

In the following we will shortly describe the file format of the input files to two programs.

Input format for "best_match"-Tool Two label list files are needed as input for "best_match"-Tool. These files have to be text files, which have as content the labels of a clustering. Each line of such input file must be one label of one cluster of the clustering. The number of the lines is also the number of clusters at this clustering. At each label, the words, which are members of a label, have a word statistic in parenthesis at the end and are separated by commas. The following line is an example for a label at such label list file: *"secur(0.54),statist(0.47),queri(0.31)"*.

Input format for visualisation tool The visualisation tool needs also text files as input format. How many input files are needed depends on the number of time periods T, which we will monitor with the visualisation tool. The number of input files is $T - 1$. These input files must have a number followed by an arrow and a second number at each line. This is the same format as the short result list used to show the matching labels found during the matching section in the result box. For example one line at the input

file looks like $3 \rightarrow 4$. This means that label 3 of period 1 match to label 4 at period two.

Summary In this chapter we shortly introduced our Theme Monitoring Tool TheMoT for use very easy our developed framework. TheMoT consists of two parts, the ”best_match()”-tool and the visualisation tool for present the monitoring results over the observed time periods as a graph. Also shortly we gave a description of the input files to both tools and show that both tools are not resource intensive.

6 Conclusion and Outlook

6.1 Summary

Monitoring topics on document archives, which change over time is an important subject in research and business. For instance new topics arise in research archives over publications, like data mining or information retrieval. Similar examples can be given at business document archives.

For this problem of monitoring such changes we developed an algorithm, the ”ThemeFinder”. It is based only on the labels of a clustering of a time period of documents of a document archive and do not force the user to take a detailed view into the clusters itself.

The ”ThemeFinder” takes as input two label files from the clusterings of the different time points and try to find out for each label of the first label file the best corresponding label at the second label file. This search is based only on the word members of each label and the statistics of this words.

As add-on or feature can be seen the possibility to reduce the creation of feature spaces during the monitoring process, specially the vectorisation of the documents at each time point. At first at each period it will be used the feature space of the previous period to vectorise the documents and then start the process of clustering, label creation and monitoring with the ”ThemeFinder”. The ”ThemeFinder” have defined different quality checks of the labelling and monitoring process to define the used feature space as good feature space for this time period. Only if the used feature space of the previous period is not a good feature space after the quality checks, the actual feature space of the period will be created and used for the clustering and monitoring process.

This algorithm is also implemented in a prototype, the Theme Monitoring Tool. This tool gives the users a fast overview about the results of this monitoring process, specially about the finding of the corresponding labels and present in coloured way the evolution of the cluster labels.

6.2 Results of this work

The main idea of this work is to develope an algorithm for topic monitoring over changing document archives. For this idea we developed the ”ThemeFinder” and try to show in several experiments the functionality and usability of this algorithm. The main results can be summarised as following:

- The ”ThemeFinder” can be used for monitoring tasks at non accumulated datasets, shown in Chapter 4.2
- equally well at accumulated datasets (Chapter 4.4)
- The cluster monitoring via cluster label monitoring with ”ThemeFinder” produce similar results as the more complex framework FOCUS from Ganti et al [Ganti 99b]
- The monitoring process is independent of the used clustering algorithm and works on different clustering algorithms
- The monitoring process is independent of the used labelling method and works with more than one labelling methods, (except the χ^2 method in combination with the feature space adjustment feature)
- The feature space adjustment feature save time and costs for the feature space creation and the monitoring process with ”ThemeFinder” only loose infinite on quality

This shows that our developed algorithm ”ThemeFinder” is a good method for topic monitoring over time and via the implementation of this algorithm at the tool Theme Monitoring Tool the user can use this framework very fast and easy.

”ThemeFinder” gives the user the possibility to get very fast an overview about changes at the document archives and so it is ideal for decision maker positions.

6.3 Future work

Here we will shortly give an idea about possible future tasks at our development. We can split the task in two main points, on the one hand the development of the implementation of the algorithm, the tool Theme Monitoring Tool. On the other hand we have also some ideas for future research at the algorithm or for extensions of it.

6.3.1 Ideas for Theme Monitoring Tool

As first next step at this development we can image to finish the implementation of the algorithm in the Theme Monitoring Tool to make it complete useful for the end users. An extension can be to implement an interface to the Theme Monitoring Tool so that the user can select all label files of all periods and then the user get a detailed list over the changes at the labels over the whole time period. This means a combination of the ”best_match()”-tool and the visualisation tool. This step is already in progress.

Another extension depends on the data format, it can be usefully to create a PMML interface to the Theme Monitoring Tool. PMML is the shortcut for ”Predictive Model Markup Language” a standard for data mining data and results exchange defined by the data mining group (www.dmg.org) This make it possible to use other data mining tools to create the clusterings and labels over the time and use the ”ThemeFinder” for the

monitoring process. This would be reduce the costs for transform the label results to the TheMoT tool.

6.3.2 Ideas for the ”ThemeFinder”

The future work at the algorithm ”ThemeFinder” can be to extends the methods on classification and create class labels about the content of a class as representation of the classes and monitoring their changes for example.

Another idea is to extend the monitoring process to completely changed feature spaces for example (without the feature space adjustment feature of course). It means we use at each period the actual feature space but the union of both feature spaces is extremely small or an empty set. This will lead to an empty set of corresponding labels from one period to the next. The question could be how to handle such cases. This question also could be used to extend the method by an automatic feature space adjustment method or other ideas, how the evaluation of a feature space can be effected.

Bibliography

[Aggarwal 05] Charu Aggarwal. *On Change Diagnosis in Evolving Data Streams*. IEEE TKDE, vol. 17, no. 5, pages 587–600, May 2005.

[Aggarwal 06] Charu C. Aggarwal & Philip S. Yu. *A Framework for Clustering Massive Text and Categorical Data Streams*. In Proceedings of the SIAM conference on Data Mining 2006. ACM, April 2006.

[Aggarwal 07] Charu C. Aggarwal, editeur. Data Streams, Models and Algorithms. Springer, 2007.

[Allan 02] J. Allan. Introduction to topic detection and tracking. Kluwer Academic Publishers, 2002.

[AlSumait 08] L. AlSumait, D. Barbara & C. Domeniconi. *On-line LDA: adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking*. In ICDM, 2008.

[Anaya-Sanchez 10] H. Anaya-Sanchez, A. Pons-Porrata & R. Berlang-Llavori. *A document clustering algorithm for discovering and describing topics*. Pattern Recognition Letters, vol. 31, pages 502–510, 2010.

[Bade 07] Korinna Bade, Marcel Hermkes & Andreas Nürnberger. *User oriented hierarchical information organization and retrieval*. In Lecture notes in computer science, Machine learning: ECML 2007, pages 518–526. Springer Berlin, 2007.

[Banerjee 07] Arindam Banerjee & Sugato Basu. *Topic Models over Text Streams: A Study of Batch and Online Unsupervised Learning*. In SDM, 2007.

[Baron 03] Steffan Baron, Myra Spiliopoulou & Oliver Günther. *Efficient Monitoring of Patterns in Data Mining Environments*. In Leonid A. Kalinichenko, Rainer Manthey, Bernhard Thalheim & Uwe Wloka, editeurs, Proceedings of Advances in Databases and Information Systems, 7th East European Conference, ADBIS 2003, volume 2798 of *Lecture Notes in Computer Science*, pages 253–265. Springer, 2003.

[Bartolini 04] Ilaria Bartolini, Paolo Ciaccia, Irene Ntoutsi, Marco Patella & Yannis Theodoridis. *A unified and flexible framework for comparing simple and complex patterns*. In Proceedings of Knowledge Discovery in databases: PKDD 2004, pages 496–499, September 2004.

[Berkhin 02] Pavel Berkhin. *Survey Of Clustering Data Mining Techniques*. Rapport technique, Accrue Software Inc., July 2002.

[Blei 03] David M. Blei, Andrew Y. Ng & Michael I. Jordan. *Latent Dirichlet Allocation*. J. Mach. Learn. Res., vol. 3, pages 993–1022, 2003.

[Blei 06] David M. Blei & John D. Lafferty. *Dynamic topic models*. In William W. Cohen & Andrew Moore, editeurs, Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006, volume 148 of *ACM International Conference Proceeding Series*, pages 113–120. ACM, 2006.

[Chung 05] Seokkyung Chung, Jongeun Jun & Dennis McLeod. Encyclopedia of data warehousing and mining, chapitre Incremental Mining from News Streams, pages 606 – 610. Idea Group Publishing, 2005.

[Deerwester 90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer & Richard Harshman. *Indexing by Latent Semantic Analysis*. Journal of the American Society of Information Science, vol. 44, no. 6, pages 391–407, 1990.

[Dries 09] Anton Dries & Ulrich Rückert. *Adaptive concept drift detection*. Statistical Analysis and Data Mining, vol. 2, no. 5-6, pages 311–327, 2009.

[Durfee 08] Antonina Durfee. Handbook of research on public information technology, chapitre Text Mining, pages 592 – 603. Idea Group Publishing, 2008.

[Dvorský 04] Jiri Dvorský, Jan Martinovic & Václav Snásel. *Query Expansion and Evolution of Topic in Information Retrieval Systems*. In Václav Snásel, Jaroslav Pokorný & Karel Richta, editeurs, Proceedings of the Dateso 2004 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, April 14-16, 2004, volume 98 of *CEUR Workshop Proceedings*, pages 117–127. CEUR-WS.org, 2004.

[Ester 96] M. Ester, H.-P. Kriegel, J. Sander & X. Xu. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with*

Noise. In Proc. 2nd int. Conf. on Knowledge Discovery and Data Mining (KDD 96), Portland, Oregon, USA, 1996. AAAI Press.

[Ester 98] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer & Xiaowei Xu. *Incremental Clustering for Mining in a Data Warehousing Environment*. In Ashish Gupta, Oded Shmueli & Jennifer Widom, editeurs, VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA, pages 323–333. Morgan Kaufmann, 1998.

[Ester 00] Martin Ester & Jörg Sander. Knowledge Discovery in Databases. Techniken und Anwendungen. Springer Verlag, 27 September 2000. ISBN 3540673288.

[Fayyad 96] Usama M. Fayyad, Gregory Piatetsky-Shapiro & Padhraic Smyth. Advances in Knowledge Discovery and Data Mining, chapitre From Data Mining to Knowledge Discovery: An Overview, pages 1–36. The MIT Press, 1996.

[Feldman 07] Ronen Feldman & James Sanger. The text mining handbook - advanced approaches in analyzing unstructured data. Cambridge University Press, 2007.

[Fung 03] B. Fung, K. Wang & E. Martin. *Hierarchical document clustering using frequent itemsets*. In D. Barbara & C. Kamath, editeurs, Proceedings of the Third SIAM Internat. Conf. on Data Mining, pages 59–70, 2003.

[Gama 10] Joao Gama, editeur. Knowledge Discovery from Data Streams. CRC Press, 2010.

[Ganti 99a] Venkatesh Ganti, Johannes Gehrke & Raghu Ramakrishnan. *CACTUS: Clustering Categorical Data Using Summaries*. In Surajit Chaudhuri & David Madigan, editeurs, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 73–83, N.Y., August 1999. ACM Press.

[Ganti 99b] Venkatesh Ganti, Johannes Gehrke & Raghu Ramakrishnan. *A Framework for Measuring Changes in Data Characteristics*. In ACM, editeur, Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems: PODS 1999: Philadelphia, Pennsylvania, May 31–June 2, 1999, pages 126–137, New York, NY 10036, USA, 1999. ACM Press.

[Ganti 00] V. Ganti, J. Gehrke & R. Ramakrishnan. *DEMON: Mining and Monitoring Evolving Data*. In 16th International Conference on Data Engineering (ICDE' 00), pages 439–448, Washington - Brussels - Tokyo, March 2000. IEEE.

[Gentle 04] James E. Gentle, Wolfgang Härdle & Yuichi Mori. Handbook of computational statistics - concepts and methods. Springer Verlag heidelberg, germany, 2004.

[Gil-Garca 10] Reynaldo Gil-Garca & Aurora Pons-Porrata. *Dynamic hierachical algorithms for document clustering*. Pattern Recognition Letters, vol. 31, pages 469–477, 2010.

[Giudici 03] Paolo Giudici. Applied data mining - statistical methods for business and industry. John Wiley and Sons, 2003.

[Graubitz 01] Henner Graubitz, Myra Spiliopoulou & Karsten Winkler. *The DIAsDEM framework for converting domain-specific texts into XML documents with data mining techniques*. In Proc. of the 1st IEEE Intl. Conf. on Data Mining, pages 171–178, San Jose, CA, Nov. 2001. IEEE.

[Greene 07] Derek Greene. *A State-of-the-Art Toolkit for Document Clustering*. PhD thesis, University of Dublin, Trinity College, March 2007.

[Griffiths 04] T.L. Griffiths & M. Steyvers. *Finding scientific topics*. PNAS, vol. 101, no. suppl_1, pages 5228–5235, 2004.

[Guha 03] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani & Liadan O'Callaghan. *Clustering Data Streams: Theory and Practice*. IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 3, pages 515–528, May 2003.

[Hand 01] David Hand, Heikki Mannila & Padhraic Smyth. Principles of data mining. MIT Press, 2001.

[Hofmann 01] T. Hofmann. *Unsupervised Learning by Probabilistic Latent Semantic Analysis*. Machine Learning, vol. 42, no. 1, pages 177–196, 2001.

[Huang 05] Xiangji Huang. Encyclopedia of data warehousing and mining, chapitre Cluster Analysis Algorithms, pages 159 – 164. Idea Group Publishing, 2005.

[Ipeirotis 05] Panagiotis G. Ipeirotis, Alexandros Ntoulas, Junghoo Cho & Luis Gravano. *Modeling and Managing Content Changes in Text*

Databases. In Proceedings of the 20th IEEE International COnference on Data Engineering (ICDE'05), pages 606–617. IEEE Computer Society, 2005.

[Karypis 00a] George Karypis & Eui-Hong (Sam) Han. *Fast Supervised Dimensionality Reduction Algorithm with Apllications to Document Categorization & Retrieval*. In Proceedings of CIKM-00, pages 12–19. ACM Press, New York, US, 2000.

[Karypis 00b] George Karypis, Michael Steinbach & Vipin Kumar. *A Comparison of Document Clustering Techniques*. TextMining Workshop at KDD2000, May 2000.

[Kim 05] Han-Joon Kim. Encyclopedia of data warehousing and mining, chapitre Text Mining Methods for Hierarchical Document Indexing, pages 1113–1119. Idea Group Publishing, 2005.

[Kontostathis 03] April Kontostathis, Leon Galitsky, William M. Pottenger, Soma Roy & Daniel J. Phelps. A survey of emerging trend detection in textual data mining. Springer Verlag, 2003.

[Kontostathis 04] April Kontostathis, Lars E. Holzman & William M. Pottenger. *Use of term clusters for emerging trend detection*. Rapport technique, 2004.

[Kroeze 05] Jan H. Kroeze. Encyclopedia of data warehousing and mining, chapitre Discovery Unknown Patterns in Free Text, pages 382 – 386. Idea Group Publishing, 2005.

[Kruschwitz 05] Udo Kruschwitz. Intelligent document retrieval. Springer Verlag, 2005.

[Lee 99] D. D. Lee & H. S. Seung. *Learning the parts of objects by non-negative matrix factorization.* Nature, vol. 401, no. 6755, pages 788–791, October 1999.

[Leskovec 09] Jure Leskovec, Lars Backstrom & Jon Kleinberg. *Meme-tracking and the dynamics of the news cycle*. In KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 497–506, New York, NY, USA, 2009. ACM.

[Li 08] Y. Li, S.M. Chung & J.D. Holt. *Text document clustering based on frequent word meaning sequences*. Data and Knowledge Engineering, vol. 64, no. 1, pages 381–404, 2008.

[Ling 08] Xu Ling, Qiaozhu Mei, ChengXiang Zhai & Bruce Schatz. *Mining multi-faceted overviews of arbitrary topics in a text collection*. In KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 497–505, New York, NY, USA, 2008. ACM.

[Mei 05] Qiaizhu Mei & ChengXiang Zhai. *Discovering Evolutionary Theme Patterns from Text - An Exploration of Temporal Text Mining*. In Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 198–207, Chicago, Illinois, USA, August 2005. ACM Press.

[Mladenic 05] Dunja Mladenic. Encyclopedia of data warehousing and mining, chapitre Text Mining - Machine Learning on Documents, pages 1109–1112. Idea Group Publishing, 2005.

[Moens 00] Marie-Francine Moens. Automatic indexing and abstracting of document texts. Kluwer Academic Publisher, 2000.

[Moringa 04] Satoshi Moringa & Kenji Yamanishi. *Tracking Dynamics of Topic Trends Using a Finite Mixture Model*. In Ronny Kohavi, Johannes Gehrke, William DuMouchel & Joydeep Ghosh, editeurs, Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining, pages 811–816. ACM Press New York, NY, USA, August 2004.

[Nasukawa 01] T. Nasukawa & T. Nagano. *Text analysis and knowledge mining system*. IBM systems Journal, vol. 40, no. 4, pages 967–984, 2001.

[Neill 05] Daniel Neill, Andrew Moore, Maheshkumar Sabhnani & Kenny Daniel. *Detection of Emerging Space-Time Clusters*. In Proceedings of 11th ACM SIGKDD International Conference on Knowlegde Discovery and Data Mining (KDD 05), pages 218–227, Chicago, IL, Aug. 2005.

[Petersohn 05] Helge Petersohn. Data mining - verfahren, prozesse, anwednungsarchitekturen. Oldenburg Verlag München Wien, 2005.

[Popescul 00] Alexandrin Popescul & Lyle H. Ungar. *Automatic Labeling of Document Clusters*, 2000.

[Pottenger 01] W.M. Pottenger & T. Yang. *Detecting Emerging Concepts in Textual Data Mining*. In Proc. of First SIAM Int. Conf. on Data Mining (SDM'01), Aug. 2001.

[Rand 71] W.M. Rand. *Objective criteria for the evaluation of clustering methods*. Journal of the American Statistical Association, vol. 66, pages 846–850, 1971.

[Roiger 03] Richard J. Roiger & Michael W. Geatz. Data mining - a tutorial-based primer. Addison-Wesley, 2003.

[Roy 02] S. Roy, D. Gevry & W.M. Pottenger. *Methodologies for Trend Detection in Textual Data Mining*. In Proc. of TextMine'02 (Workshop in the Second SIAM Int. Conf. on Data Mining - SDM'02), Apr. 2002.

[Salton 89] Gerard Salton. Automatic Text Processing, The transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley Publishing Company, 1989.

[Sanfilippo 04] Antonio Sanfilippo, Gus Calapristi, Vernon Crow, Beth Hetzler & Alan Turner. *Meaningful Clusters*. In Proceedings of LREC04, May 2004.

[Schult 06a] Rene Schult & Myra Spiliopoulou. *Discovering emerging topics in unlabelled text collections*. In Advances in Databases and Information Systems, 10th East-European Conference,(ADBIS'2006), pages 353–366. Springer Verlag, September 2006.

[Schult 06b] Rene Schult & Myra Spiliopoulou. *Expanding the Taxonomies of Bibliographic Archives with Persistent Long-Term Themes*. In Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC'06), April 2006.

[Schult 08] Rene Schult & Myra Spiliopoulou. *Market Monitoring with Topic Evolution Monitoring*. Datenbank Spektrum, vol. 25/2008, pages 23 – 30, 2008.

[Song 05] Xiaodan Song, Ching-Yung Lin, Belle L. Tseng & Ming-Ting Sun. *Modeling and predicting personal information dissemination behavior*. In SIGKDD, pages 479–488. ACM, 2005.

[Spiliopoulou 04] Myra Spiliopoulou & Steffan Baron. *Temporal Evolution and Local Patterns*. In Katharina Morik, Jean-François Boulicaut & Arno Siebes, editeurs, Local Pattern Detection, volume 3539 of *Lecture Notes in Computer Science*, pages 190–206. Springer, 2004.

[Spiliopoulou 06] Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis & Rene Schult. *MONIC: modeling and monitoring cluster transitions*. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven & Dimitrios Gunopulos, editeurs, Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006, pages 706–711. ACM, 2006.

[Stein 04] Benno Stein & Sven Myer zu Eissen. *Topic Identification: Framework and Application*. In Proceedings of the I-KNOW'04, 4th International Conference on Knowledge Management, Graz, 2004.

[Strehl 02] A. Strehl & J. Gosh. *Cluster ensembles - a knowledge reuse framework for combining multiple partitions*. Journal of Machine Learning Research, vol. 3, pages 583–617, 2002.

[Tan 06] Pang-Ning Tan, Michael Steinbach & Vipin Kumar. Introduction to data mining. Pearson Education, 2006.

[Tasoulis 05] D.K. Tasoulis & M.N. Vrahatis. *Unsupervised clustering on dynamic data bases*. Pattern Recognition Letters, vol. 26, pages 2116–2127, 2005.

[Wang 06] Xuerui Wang & Andrew McCallum. *Topics over time: a non-Markov continuous-time model of topical trends*. In SIGKDD, pages 424–433. ACM, 2006.

[Whitney 09] Paul Whitney, Dave Engel & Nick Cramer. *Mining for surprise events within text streams*. In Proceedings of the ninth SIAM International Conference on Data Mining, pages 617–627. Society for Industrial and Applied Mathematics, 2009.

[Widyantoro 05] D.H. Widyantoro & J. Yen. *Relevant Data Expansion for Learning Concept Drift from Sparsely Labeled Data*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 3, pages 401–412, 2005.

[Winkler 02] Karsten Winkler & Myra Spiliopoulou. *Structuring Domain-Specific Text Archives by Deriving a Probabilistic XML DTD*. In 6th European Conf. on Principles and Practice of Knowledge Discovery in Databases, PKDD'02, pages 461–474, Helsinki, Finland, Aug. 2002.

[Yang 97] Y. Yang & J.O. Pedersen. *A comparative study on feature selection in text categorization*. In D.H. Fisher, editeur, Proceedings of 14th International Conference on Machine Learning (ICML'97, pages 412 – 420, San francisco, US, Nashville, 1997. Morgan Kaufmann Publishers.

[Yang 05] Hui Yang, Srinivasan Parthasarathy & Sameep Mehta. *A generalized framework for mining spatio-temporal patterns in scientific data*. In Robert Grossman, Roberto J. Bayardo & Kristin P. Bennett, editeurs, Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005, pages 716–721. ACM, 2005.

[Zhao 02] Ying Zhao & George Karypis. *Criterion Functions for Document Clustering*. Rapport technique 01-40, Army HPC research Center, 21 February 2002.

[Zhong 05a] Shi Zhong. *Efficient online spherical k-means clustering*. In Proceedings IEEE international joint conference on neural networks, Montreal, Canada, Aug 2005.

[Zhong 05b] Shi Zhong. *Efficient streaming text clustering.* Neural Networks, vol. 18, no. 5-6, pages 790–798, 2005.