Any-time clustering of high frequency news streams

Fabian Moerchen
Siemens Corporate Research
755 College Road East
08540 Princeton, NJ, USA
001-609-734-3529
fabian.moerchen@siemens.com

Klaus Brinker
Siemens Corporate Research
755 College Road East
08540 Princeton, NJ, USA
001-609-734-3312
klaus.brinker@gmail.com

Claus Neubauer
Siemens Corporate Research
755 College Road East
08540 Princeton, NJ, USA
001-609-734-6567
claus.neubauer@siemens.com

ABSTRACT
We describe a large scale system for clustering a stream of news articles that was developed as part of the Geospace & Media Tool (GMT). The GMT integrates the news feed with geospatial, census, and human network information to provide a research tool for members of Congress and their staffs. News articles covering the same event are summarized for the user through the clustering component. The clustering result is available to the user at any time without additional on-demand clustering steps. The documents are grouped into clusters on-the-fly without any assumptions on the number of clusters and without retrieving previous documents. High efficiency is achieved by utilizing locality sensitive hashing (LSH) as a means to determine a small set of candidate clusters for each document. This way a large number of clusters can be considered while keeping the number of expensive document to cluster comparisons low. Our experiments with the system reveal interesting aspects of large-scale text processing in general and news clustering in particular. We demonstrate how the LSH based approximation achieves a large speedup at the cost of only few and small errors. On a high-frequency benchmark data set a clustering quality comparable to one of the best non-streaming document clustering algorithms is obtained.

Categories and Subject Descriptors
H.4 **[Information Systems Applications]***: Miscellaneous*

General Terms
Algorithms, Performance.

Keywords
text mining, clustering, data streams.

1. INTRODUCTION
The automated processing of large amounts of text is an important tool in knowledge management [9, 18]. Classification and clustering of text documents can help to structure a document collection and make it more accessible. At a coarse level an assignment to topics or categories can help to navigate a corpus. If some of the documents are already assigned to topics, a supervised classification approach can help to label the remaining documents and any new documents [34, 43, 29].

Otherwise unsupervised clustering can help to discover the underlying topical structure of existing text collections [15, 35, 52]. In many applications the document collection is dynamic [5, 48, 30] in the sense that new documents are continuously being added to the database and need to be processed. In this scenario both supervised and unsupervised methods need to be able to cope with new topics in the text that do not fit the previous classification or cluster model [46, 20].

Consider news articles from newspapers and news agencies that form a stream of text documents. For a given event different news sources publish similar articles and even the same news source covers a developing story over several days. Clustering can be used to aggregate the news articles that cover the same story and offer the user a better overview of the current events [5, 48, 30]. For each cluster a summary can be generated from using headlines and content of representative articles and the most relevant keywords. Additional articles from the cluster can be displayed on demand to provide more details. A well known implementation of such a news aggregation system is Google News (http://news.google.com).

We describe a large scale system for clustering a stream of news articles that is a core component of Geospace & Media Tool (GMT) developed in cooperation with the Parsons Institute for Information Mapping (PIIM), The New School, NY. The GMT integrates the news feed with geospatial, census, and human network information to provide a research tool for the members of Congress and their staffs. We describe the requirements of the text clustering component within this application and describe an efficient solution. We report the results of extensive experiments regarding the trade-off between speed and quality, the handling of dynamic content, and the merits of using meta-data to improve the clustering quality. The same system could be applied to cluster Blog entries, emails, customer service requests, medical reports, and similar potentially high rate text streams. Section 2 briefly describes the different components and use cases of the GMT and translates this into requirements for the clustering of new articles. In Section 3 we review related work on text clustering. The data set used in evaluating the system is described in Section 4. Sections 5-6 describe our online text clustering system. The evaluation with news articles in Section 7 shows the high

DMCS Workshop, KDD'07, August 12–15, 2007, San Jose, CA, USA.

clustering quality and discusses the influence of some key parameters on the performance. The results and lessons learned are discussed in Section 8 and the achievements are summarized in Section 9.

2. GEOSPACE & MEDIA TOOL

The GMT is a government funded tool developed under the lead of the Parsons Institute for Information Mapping (PIIM), The New School, NY. It integrates a stream of news articles provided by Factiva (http://www.factiva.com) with geospatial, census, and human network information to provide a research tool for the United States' senators and their staff. The news articles are processed by a customized entity extraction module that combines off-the-shelve software for named entity and geo location detection with algorithms for disambiguation. The articles and the detected entities are then processed by the clustering component described in detail in this study. The news articles and clusters are stored in a relational database system for access by the web-based GMT client. The client lists the currently active top stories and supports user-defined keyword and location searches. The ranking of top stories and search results utilizes statistics pre-computed during the clustering. For each news cluster several representative headlines and keywords are displayed. The extracted locations of the news articles and clusters are used to display them on a zoomable map. Census data provided by ESRI (http://www.esri.com) can be displayed on top of the map to provide context to news stories. The connections between extracted people and organizations can be explored with network displays that are complemented with biographical and contact information.

The clustering component of the GMT serves to summarize news articles about one particular event or topic for the users. The grouping of similar articles eases the browsing of the vast amount of content. The ranking helps in determining the currently most important topics. The users of the GMT are further interested in small stories with local scope. The geo-location of news articles helps in searching such articles via the interactive map or by specifying regions along with tops keywords. For the clustering it means that all news articles need to be clustered, not only articles about the major topics. Existing systems like Google News only display top stories and it is unclear what happens to the rest of the news. Preliminary user tests resulted in positive feedback about the usefulness of the system to congressional staff to solve the daily task of researching various current topics.

Each resulting news cluster should contain as many articles about an event as possible (high recall) and only contain only few articles with low relevance (high precision). The system needs to find a tradeoff between these two quality measures for both large clusters and small clusters simultaneously.

The news stream used in the GMT contains about 50k-100k articles per day. The news articles need to be clustered as fast as possible to provide timely information to the users. With such a high rate of incoming articles clustering of the complete data set will quickly become infeasible. Even clustering only the articles from the last few days will be demanding and more importantly useless, because the result will be outdated when it is available. Data stream clustering techniques are needed [24, 2, 22, 3, 1] to continuously process the feed. The data stream model is commonly characterized such that "the data elements in the stream arrive online", "the system has no control over the order [...]", "data streams are potentially unbounded in size" [6] and there is only one chance of fast access for each data element. For data mining in general [17] and clustering in particular [8] this has been translated into certain requirements.

In summary, we identified the following conditions as being crucial for the clustering the news stream in the GMT:

- Process the stream in a single pass using a small constant time per record and only a fixed amount of main memory [17].
- Process all documents [8], i.e., do not use load-shedding or outlier removal.
- Create a clustering similar in quality to non-streaming algorithms [17].
- Make the clustering available at any point in time [17].
- Do not make assumptions about the number of clusters.
- Dynamically adjust to changing content.

We developed a new solution for this problem because no previously proposed method met the above requirements sufficiently (see Section 3). Even though our method is quite simple it achieves excellent clustering quality in an application to news processing.

3. RELATED WORK

Introductions to text clustering can be found in [35, 9, 52, 18]. An analysis of the efficiency and quality of various building blocks from the popular k-Means and Scatter- Gather [15] algorithms for large datasets is done in [31]. In [35] the bi-secting k-Means algorithm performed better than k-means and hierarchical algorithms. The extensive evaluation of hierarchical clustering algorithms concludes that "partitional methods are suitable for producing flat and hierarchical clustering solutions for document datasets effectively and efficiently" [52].

Two well known incremental hierarchical clustering algorithms are BIRCH [49] for numerical data and COBWEB [19] for categorical data. In [39] a variant of COBWEB for text documents is described. The algorithms can update the current clustering model upon arrival of new data points, but they are not well suited for data stream processing [8]. The time and memory requirements of COBWEB can degrade because the internal tree structure is not balanced. BIRCH is designed to use slow secondary memory for the cluster model. Even if the cluster model can be kept in memory completely by using pruning techniques it does not necessarily correspond to a natural cluster structure, a final clustering of the leaf nodes is required [49].

The incremental competitive learning algorithm of [7] is targeted toward text documents and produces a flat clustering. Exactly k clusters of about the same size clusters are created. For a detailed clustering of a collection very different cluster sizes are more desirable as noted for news articles in [45]. Here, a cluster model of a news archive is updated daily in a batch process, thus not making the result available in real time as new articles arrive. In [50, 51] self splitting competitive learning is advocated to free the user from choosing the number of clusters. In order to assign the vectors of a splitted cluster to the new sub-clusters the vectors of

previous data points are needed, violating the one pass requirement.

Clustering of text streams has also been used for retrospective theme detection [36] and topic detection and tracking in the TDT workshops [48]. Topic detection aims to find the first incoming document of a new topic. Here, comparing new documents to all previous documents has worked better than comparing them to clusters [10]. Topic tracking aims at discovering all document that belong to the topic given a certain number of example documents. Topic tracking can be done in retrospective or online [5, 48] with supervised and unsupervised methods [47, 21]. Unsupervised online clustering corresponds to combined topic detection and topic tracking given the first detected document. For the retrospective analysis several clustering algorithms are compared in [48]. Hierarchical clustering performed best, but online single-pass clustering was almost as good. This was explained with the temporal proximity of topics in the data set. In contrast to our requirements some topic tracking systems do not assign all documents to clusters [10].

In [16] the very fast k-Means (VFKM) algorithm is proposed for huge datasets, but it requires several passes over the data using a sample with increasing size in each pass. A one-pass approximation to k-Medians is described in [37, 24]. The data points are clustered in batches to obtain a large number of weighted medians which are successively re-clustered until only k centers remain. The number of clusters k needs to be specified at least roughly and the clustering result is not available at any time. Whenever a clustering of the data observed so far is desired, the clustering up to the final level needs to be initiated.

The approach proposed in [2, 3, 1] does not make any assumptions on the number of clusters, but also divides the process into an online and offline component. The online part keeps a collection of so-called micro-clusters with sufficient statistics and temporal information about the data points assigned to it up to date. At certain time points a snapshot of these clusters is saved. The offline part constructs the final clustering from the stored micro-clusters for a specified time horizon on demand. This step is potentially very expensive for large time horizons. A density based micro-clustering is described in [12]. In [28] the data stream is first segmented by a change point detection algorithm. Each time a segment boundary is found the preceding segment is clustered. This can potentially cause a large delay between the time a data point is observed and when it is clustered.

4. DATA SET

Our text clustering system was developed to process the stream of news from many (online) news sources collected by the news provider Factiva. Depending on the number of original sources about 50k-200k articles per day need to be processed. The goal was to provide an efficient way to cluster the news articles at a story level to support the browsing of the articles. This is in contrast with text categorization [34] or topic tracking (e.g. [5]) where broad categories or only major news stories are of interest. The ground truth data available with commonly used benchmark datasets like Reuters-21578 [33] or RCV1 [32] corresponds to (coarse) topic categories and not individual news stories. The TDT datasets contain labeled news stories but the data is collected over several months resulting in relatively low daily and hourly rates of articles that do not represent our target stream well. Moreover, certain datasets like TDT2 [14] include structurally diverse news items, such as radio and TV broadcasts, or have been compiled to benchmark different tasks, such as supervised adaptive topic tracking and multilevel hierarchical topic detection (TDT56). The labeling procedure of the TDT datasets is further biased towards larger stories [48].

We therefore collected the daily news articles from 02/16 to 02/28 in 2006 and labeled 80 news stories of various sizes as our ground truth for evaluation. Several large stories were identified monitoring the websites of major newspapers during the same time period. Smaller local stories were found by filtering the complete data with keyword queries like “Phoenix” and performing an initial very fine grained clustering on this subset. For each story a thorough semiautomatic labeling process was performed involving the following steps executed repeatedly as necessary:

- Search the database for more articles containing important keywords present in the current story.
- Rank the selected articles by their similarity (see Section 5) to the center of the current selection or the closest article in the current selection to help distinguishing relevant and irrelevant articles.
- Search the database for more articles with high similarity to any selected article.
- Rank unselected candidate articles by their similarity to the closest article in the current selection to add very similar articles.

For borderline articles the decision about what was included in a story was checked by at least two persons. The 80 true stories were split into a training part used in optimizing the crucial system parameters and an independent test part to evaluate them. For each data set we added about 1000 unlabeled articles per day randomly selected from the stream. Some statistics of the final data sets are shown in Table 1.

Table 1: Characteristics of benchmark data sets derived from high density text stream

Data set	Articles		Articles per story		
	labeled	unlabeled	min	median	max
Train	4168	12416	7	34	682
Test	3674	12371	4	40	643

Large stories included a discussion about the hunting accident of Cheney, the resignation of the president of Harvard University and calls for the closing of Guantanamo. Typical medium sized stories were the delays at Delphi and the hostages abducted from an oil platform in Nigeria. Among the smallest stories were the decision about Measure 37 in Oregon and a gunman in Phoenix. Several stories were specifically selected to be very similar, e.g., patent issues of Blackberry and of Adidas vs. Nike.

5. PREPROCESSING

The incoming documents are preprocessed with the standard text mining chain of methods [40]. First all words from a list of stop words are removed. For our news application the list included the names of several big news agencies. We further removed all Internet links and Email addresses. From the remaining words a

list of lower case word stems is generated with Porter's stemming algorithm [38]. The frequencies of the word stems are saved for each document.

For the news application the names of locations, persons, and organizations were extracted from the original (un-stemmed) text and saved separately with their occurrence frequencies. Each article was further assigned a list of subject categories similar to the RCV1 data. Each category code was treated as a word stem. The importance of this meta-information and of words from the headline and the abstract can optionally be emphasized by artificially increasing their frequencies.

Each word stem frequency is then mapped to a numerical value with the incremental TFIDF (Term Frequency Inverse Document Frequency) scheme of [11] as used in many topic detection approaches [48, 4, 10]:

$$\hat{\mathrm{TF}} = \frac{\mathrm{TF}}{\mathrm{TF} + 0.5 + \frac{1.5 \cdot \mathrm{DL}_N}{\frac{1}{N} \cdot \sum_{i=1}^{N} \mathrm{DL}_i}} \quad (1)$$

$$\mathrm{IDF} = \frac{\log\left(\frac{N+0.5}{\mathrm{DF}}\right)}{\log(N+1)} \quad (2)$$

$$\mathrm{TFIDF} = \hat{\mathrm{TF}} \cdot \mathrm{IDF} \quad (3)$$

where TF is the term frequency (how many times did the term appear in the document), *N* the number of documents processed, DL_k is length of the k-th document in words, and DF is the document frequency (in how many documents did the word appear so far). The more frequently a word appears in a document, the higher the corresponding feature value is. The document frequencies indicate how common a word stem is in the document collection. The more frequently a word appears in the corpus, the lower the corresponding feature value is.

The value of *N*, the sum of the DL_k values, and the DF for each word stem are updated incrementally as the stream is processed. In order to ensure bounded memory consumption we limit the list of word stems and their corresponding DF to a fixed number. When this limit is exceeded we discard the words stems that have not appeared in any document for the longest amount of time. This way the system can adapt to changing topics in the document stream. The most common word stems in the corpus correspond to very common terms of the English language that are not quite general enough to be used as stop words. Their frequencies stabilize quickly [11]. Many word stems with medium document frequencies correspond to currently important topics because they appeared in a significant number of documents. The word stems with very low document frequencies mostly correspond to noise (e.g. misspellings) or very small stories that have a short lifespan in the stream.

6. CLUSTERING

In order to meet our requirements we need a single pass algorithm that processes all documents with limited memory and processing time. The documents need to be assigned to clusters immediately to minimize the delay between the point in time when a document is ingested into the database and when it is available to the user as part of a cluster. We cannot make any assumptions on the number of clusters a priori. In Algorithm 6.1 we list the basic single-pass clustering algorithm [48, 44].

Each incoming document is compared to a set of candidate clusters determined by ρ. Different variants of ρ are discussed below. If the distance to the closest cluster is below the threshold T, the document is assigned to this cluster. Otherwise a new cluster containing the current document vector is created. For δ(·, ·) we use the cosine distance of a document vector to the cluster centroid where each vector is normalized to length one. For efficiency the centroid can be pruned to contain only the k largest entries [48].

Algorithm 6.1 Single pass anytime clustering algorithm.

Input:
Document vector stream $V = \{v_i | i = 1, ..., \infty\}$.
Cluster candidate selection function $\rho(\cdot, \cdot)$.
Distance function $\delta(\cdot, \cdot)$ between vectors and clusters.
Distance threshold T.

Output:
Set of clusters $C = \{c_j \subset V | j = 1, ..., \infty\}$.

$C := \emptyset$
for all $i = 1, ..., \infty$ **do**
  $\hat{C} := \rho(C, v_i) \subseteq C$
  $\hat{d} := min\{\delta(v_i, c) | c \in \hat{C}\}$
  if $\hat{d} < T$ **then**
    $\hat{c} := c \in C\ \delta(v_i, c) = \hat{d}$
    $\hat{c} := \hat{c} \cup \{v_i\}$
  else
    $C := C \cup \{\{v_i\}\}$
  end if
end for

Multiple cluster memberships can be supported with the following variation: The document is added to all clusters that are sufficiently similar and a new cluster is only created if no cluster with a distance below the threshold is found. The time needed to process a document depends on the size of the candidate cluster set C hat returned by ρ. If efficiency is not an issue we can simple choose

$$\rho_{all}(C, v) := C \quad (4)$$

i.e., compare each incoming document to all existing clusters. Clearly this will not be scalable because as the document stream progresses more and more existing clusters will need to be considered. Using only the most recent clusters from a sliding time window ensures limited memory consumption and processing time independent of the amount of previously clustered documents. Given a maximum age *A* and an age function *a (·, ·)* for clusters we can define:

$$\rho_{\mathrm{window}}(C, v) := \{c \in C | a(c, v) \leq A\} \quad (5)$$

For the age of a cluster we use the difference between the time stamp of the current document and the most recent document in the cluster, other formulations are possible.

For high density streams and long time windows this will still return too many candidates to achieve real-time processing. Let's assume we have 10k documents per day that are on average clustered into clusters of size 10. If we want to keep 1k clusters from each of the last 7 days this leaves us 1.2ms for a single comparison of a document to a cluster including the time needed for IO and preprocessing. With 50k documents per day only

0.05ms are available. We need to reduce the number of document cluster comparison to handle such high density streams together with large time windows. We propose to use locality-sensitive hashing (LSH) [27, 26, 41] to overcome this obstacle. LSH provides an index structure that can be used to determine approximate nearest neighbors, can be updated incrementally, and can deal with high dimensional data [41]. For a given vector the hash returns a (small) set of candidate clusters. These candidate clusters include the most similar cluster with high probability. Let *LSH(·, ·)* be this hash function, then we can define

$$\rho_{\text{hash}}(C, v) := \{c \in \text{LSH}(v, \rho_{\text{window}}(C, v)) \qquad (6)$$

The exhaustive search is only carried out over the much smaller set of clusters as determined by the hash function. This way large time windows can be supported without a prohibitively large increase in processing time. The hash supports online updates. New clusters are added to the hash and changed clusters are updated by removing the old cluster vector and adding the current vector. The hash returns a variable number of clusters for each request. To ensure a limited run time we keep track of the sizes of the clusters stored in the hash and use at most the M largest candidate clusters. We used an efficient variant of the hash functions described in [26]. Instead of implicitly mapping the range of each TFIDF features to several binary columns we used only one binary feature indicating whether the TFIDF value is greater than zero, i.e., whether the word stem is present in the document or not.

Some notes on implementation. The major cost of such a clustering system is the retrieval of the text from the database or files and the storage of vectors and cluster information in the database. If C hat is large the corresponding vectors cannot be kept in memory. The usage of LSH thus saves not only distance calculations but many vector retrieval operations either from a disk cache or the database. The set of relevant clusters C hat can be maintained incrementally or reinitialized in large intervals, e.g., daily. While the algorithm is formulated such that each document is processed individually, it is more efficient to process (small) batches of documents together. This will only cause a small delay in the clustering of the documents early in the batch.

7. EVALUATION

We performed a thorough evaluation of our clustering system to optimize parameters for deployment and to analyze the tradeoff between speed and quality. First we analyzed the approximation quality of our LSH based solution to find out how much scale-up one can expect for making certain errors. Next the most important parameters of the clustering algorithms were optimized on the training set and evaluated independently on the test set. We demonstrate the resulting high quality of the online system by comparing our system to a non-streaming method. Finally we performed experiments varying the size of the feature space and the size of the cluster representation to see if we can save more time and space without sacrificing the achieved quality.

7.1 Methods

During the following evaluations we fixed several parameters of the system based on prior experience to avoid a combinatorial explosion. We used 250 random hash functions based on 2 random permutations each. During clustering the hash was filled with at most 10k clusters at the beginning of each day selected from the previous 7 days based on size and age. Unless otherwise noted we used a clustering threshold of 0.76 and emphasized all meta data (see Section 7.3.1). The number of active features was at most 50k and the size of cluster vectors was not limited. The documents were processed in batches of size 100.

The clustering quality was evaluated with precision, recall, and F1 [40]. For each ground truth story we evaluated all clusters that contained at least one article of the story. Precision measures how dominant this story is in the cluster, whereas recall measures how much of the story is contained in the cluster. F1 measures a compromise between precision and recall as each can be optimized individually with a trivial solution (one cluster with all documents or one cluster per document). The cluster with the highest F1 score was selected for each story and the unweighted average of the F1 values was used to evaluate a clustering result. We chose not to use weighting by size because we want small clusters to be well represented. To de-emphasize the influence of the random number generation in the hash structure we used several repetitions for each parameter setting and report mean and standard deviation. The test data is only explicitly used in parameter optimization to avoid over fitting and the comparison to the non-streaming algorithm to ensure reproducibility. The other experiments were performed to analyze the system's behavior and give recommendations for the parameter selection.

7.2 Nearest cluster approximation

The deployed LSH speeds up the search of the cluster closest to a document vector, but it provides only an approximation. There can be cases where the nearest cluster is not part of the candidate set. We performed some experiments to evaluate the approximation quality similar to [23]. In order to investigate the trade-off between speed and error we varied the number of candidate clusters M that we use for exhaustive nearest cluster search. The more candidates we consider, the more likely the true closest cluster should be found or the smaller the possible error should be but at the same time more distance calculations and vector retrieval operations are needed.

Table 2: Quality of LSH-based nearest cluster selection.

Candidates	Error rate		Absolute Error		Speedup
	LSH	Random	LSH	Random	
250	0.39	0.93	0.0614 ± 0.1057	0.1718 ± 0.1742	20.4
500	0.31	0.89	0.0498 ± 0.0881	0.1606 ± 0.1696	11.3
750	0.26	0.87	0.0419 ± 0.0700	0.1547 ± 0.1669	8.5
1000	0.24	0.85	0.0382 ± 0.0581	0.1543 ± 0.1684	7.2
1250	0.23	0.84	0.0354 ± 0.0478	0.1514 ± 0.1669	6.6
1500	0.22	0.84	0.0345 ± 0.0450	0.1515 ± 0.1674	6.3

We compared the LSH based cluster candidate selection to an exhaustive search over all clusters in the hash as in [23] and to a random selection that picks the same number of clusters as returned by the LSH method. The results for 5 repetitions of each setting on the training data are listed in Table 2. The test data was not used, as the results are independent of the labeling. We recorded the fraction of erroneous decisions in finding the closest cluster (error rate). The standard deviation is not listed because it was 2 orders of magnitude smaller than the mean.

For all wrong decisions we calculated the mean and standard deviation of the absolute difference between the distance to the best cluster and distance to the selected cluster (absolute error) within each clustering run and list the mean values over the repetitions. To evaluate the speedup in comparison with the exhaustive optimal search we report the percentage of necessary distance calculations that also correspond to the number of vector retrieval operations. The overhead needed to maintain and query the hash is very low as the hash fits into main memory and mainly integer operations are used. We did not use wall clock timing because they are heavily influenced by the configuration of the caches and the memory management of the database and the operating system.

The LSH-based selection proved to be very effective. It leads to much fewer wrong decisions and much smaller absolute errors. With 1000 candidates less than a fourth of the decisions are wrong with a mean absolute error of only 0.04. Compared to the optimal exhaustive search a high speed can be achieved. Above 1k candidates a saturation effect is observed. This is probably due to the necessarily limited capability of the hash structure with fixed parameters. We further investigated the influence of the number of candidates on the cluster quality. The F1 values for the LSH-base approximation are shown in Figure 1. The quality increases clearly up to 500 candidates, above 1k candidates little improvement is observed. Similar results were obtained on the test set. We chose this candidate set size for further experiments.

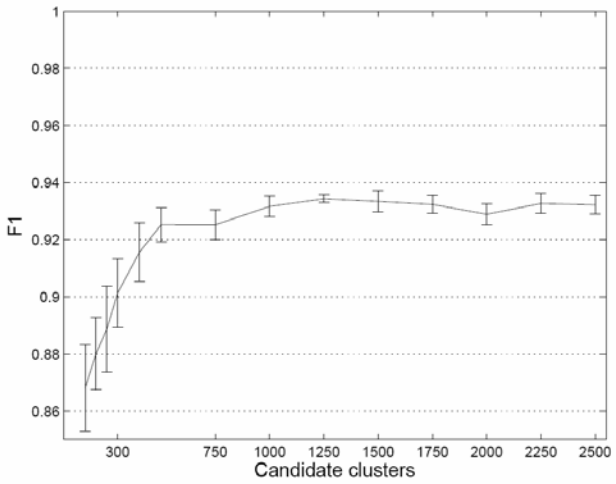

Figure 1: Cluster quality for LSH approximation with different numbers of cluster candidates.

7.3 Cluster quality

7.3.1 Clustering threshold

Optimization of the threshold parameter is crucial for achieving a good quality with single-pass clustering [48, 25]. We performed a parameter study involving the threshold and different weighting schemes for emphasis of the following meta-data: locations, persons, organizations, categories, headline, and abstract. The baseline setting does not use any meta-information. The best emphasized variant utilizes the meta-information from locations, organizations, categories, and the abstract. In order to further investigate the importance of meta-data we implemented a third weighting scheme simulating the absence of locations, persons, and organizations by removing the corresponding word stems.

Table 3: P-values for comparison of the cluster quality with a threshold of 0.76.

vs. baseline	Training	Test
Emphasized meta-data	$< 10^{-13}$	< 0.0165
Removed meta-data	$< 10^{-4}$	$< 10^{-15}$

All tests were repeated 20 times to enable an evaluation of significance with the t-test. The results for the training data are shown in Figure 2 with mean and standard deviation for several thresholds.

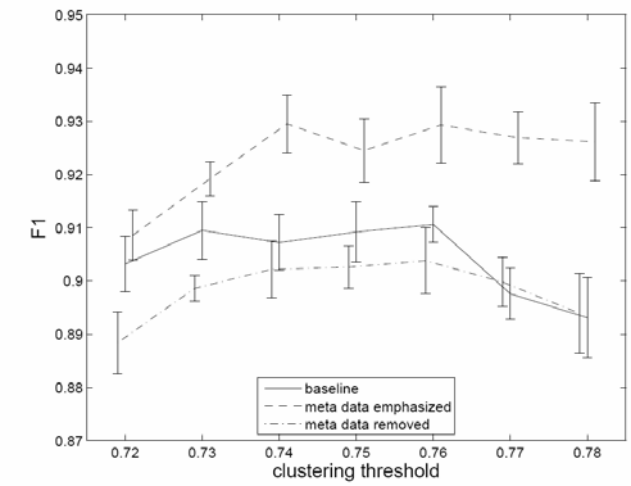

Figure 2: Cluster quality on the training data for different thresholds and meta-data weighting schemes.

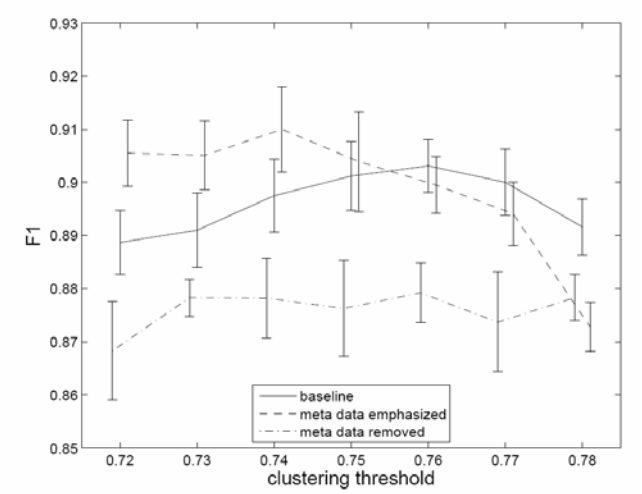

Figure 3: Cluster quality on the test data for different thresholds and meta-data weighting schemes.

The emphasis of meta data clearly improves the clustering quality on the training data. The p-values from the comparison using the best clustering threshold of each setting are shown in Table 3. This result can also be reproduced on the test data using the same threshold values as on the training data. The absolute difference in quality is smaller on the test data and for the larger thresholds (corresponding to higher recall but lower precision) it is even better than the emphasized variant. The removal of meta-data results in a significant decrease in cluster quality on both datasets as can be seen from Figures 2-3, and Table 3. It seems that the off-the-shelve text preprocessing already does a decent job in detecting which word stems are most important for the news articles. Nevertheless, emphasizing meta-data explicitly can further improve the quality significantly.

7.3.2 Comparison with hierarchical clustering

Our online clustering system performs quite well as demonstrated in the previous sections. In addition one needs to consider that even a thorough manual labeling process will never be perfect so F1 values of 100% are not to be expected. In order to estimate how much quality is lost due to the online constraints we compared our algorithm to groupwise average hierarchical clustering, one of the best offline text clustering algorithms [48, 47, 31, 25, 52]. Starting with one cluster per document the two closest clusters are merged based on the average similarity of all pairs of documents from the two clusters.

We stopped merging as soon as the number of clusters was equal to that created by the single-pass algorithm. It turned out that under these conditions the single-pass algorithm performed better than groupwise average clustering on the training data. We varied the threshold of the single-pass and thus the number of clusters for groupwise averaging until an optimum was found. The best results for each method are shown in Table 4.

Table 4: Cluster quality in comparison with offline group-wise average clustering.

Data set	Single-pass	Groupwise
Training	0.9378 ± 0.0018	0.9453 ± 0.0001
Testing	0.9091 ± 0.0073	0.9258 ± 0.0005

For both datasets the single-pass clustering achieves F1 values that are comparable to offline hierarchical clustering. The absolute differences in the F1 values are smaller than 0.01. Of course hierarchical clustering does not scale up to high frequency text streams because it requires the calculation of all pair wise distances which is quadratic in the size of the document collection.

7.3.3 Size of the feature space

An important feature of our system is the dynamic feature space. Only a limited number of word stems can be used at any time to avoid an unbounded increase in the run time of the system. A larger feature space will generally lead to a slower system and extremely large feature spaces will contain a lot of irrelevant word stems. On the other hand the feature space should not be chosen too small because then important words might be discarded if they don't occur in the currently processed document(s). If they occur again at a later point in time they will be added as a new feature with a different vector position. This can lead to errors in the assignment of a document to an existing cluster with similar documents, because such word stems will incorrectly increase the distance. We varied the maximum number of active feature from 15k to 175k and measured the number of re-appearing word stems including duplicates and the cluster

quality on the training data as shown in Figure 4 and Figure 5, respectively. Similar results were obtained on the test set.

The number of re-appearing features is very high for small numbers of active features. If at most 25k features are active at any time a feature is assigned a different vector position than for a previous occurrence more than 100k times. For 100k active features this happens only around 7k times. This is also reflected in the cluster quality that rises steeply up to 50k-75k features and does not improve past 100k features.

Both curves are certainly somewhat data set dependent, the number of unique features in this data set is about 166k. For real life high density streams we recommend to use at least 100k.

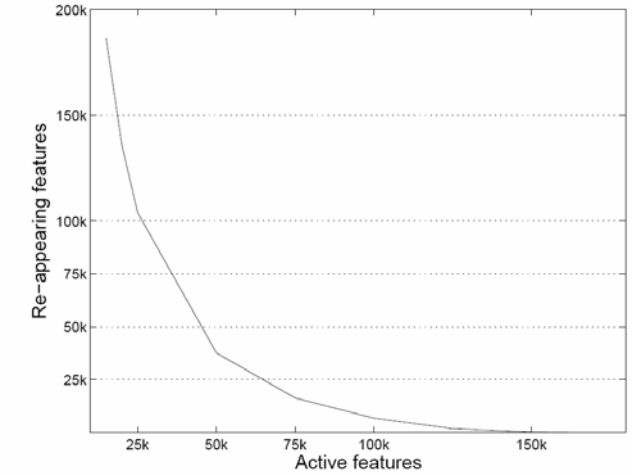

Figure 4: Number of re-appearing features for different maximum numbers of active features.

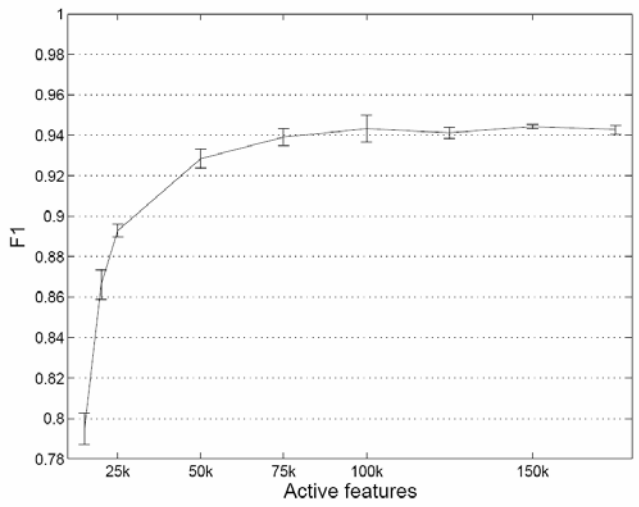

Figure 5: Cluster quality for different maximum numbers of active features.

7.3.4 Size of the cluster representation

Apart from the global limitations on the number of features, the vector based representation of each cluster can be limited. For clusters that contain many documents, the number of non-zero entries in the vector calculated as the sum of the individual documents vectors can become very large. The vector sum can be pruned by keeping only the largest d entries and setting additional values to zero [42]. This will reduce noise and increase the speed of distance and hash calculations. We varied the maximum number of features per cluster from 100 to 10k and measured the cluster quality on the training set as shown in Figure 6. Similar results were obtained on the test set.

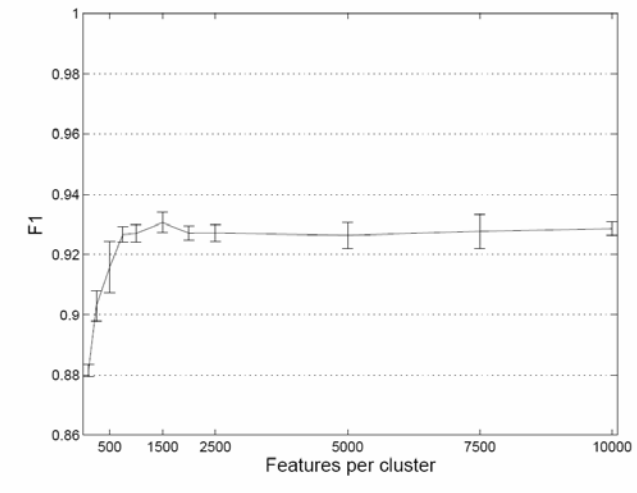

Figure 6: Cluster quality for different numbers of feature per cluster.

The clustering quality rises up to a maximum of 1.5k features per clusters. Beyond this no significant degradation could be observed. The influence of noise seems to be negligible but pruning is still worthwhile because it saves memory and processing time. We assume that the entries pruned beyond the largest 1.5k have small TFIDF values and thus do not significantly influence the clustering quality.

8. DISCUSSION

We designed a high performance text clustering system and applied it to the real world problem of news aggregation. Our main contribution is the usage of LSH to make the single-pass clustering algorithm [48, 44] scale up to high frequency text streams using a very simple hash function. Previous research used datasets with much lower density of news articles per day and or a posterior analysis of news archives. Under these conditions all clusters from a long time range can be considered for an incoming document and even iterative algorithms might be feasible. For high-frequency text streams our solution creates a good solution efficiently. We showed empirically that a very high level of cluster quality is maintained even though only a small fraction of the distance calculations and the associated retrieval of cluster vectors are needed. Our reported experiences give insight into the problems that are encountered when dealing with large-scale problems.

We store the centroid vectors of clusters from a sliding time window in the hash structure to find good candidate clusters for an incoming document vector. If memory permits, document vectors could be stored in the hash with the corresponding cluster

id to assign a document to the cluster with the closest document. For topic detection this single-link approach is of advantage [48] whereas for clustering the former group-average paradigm is reported to work better [25].

The clustering quality of the single-pass clustering algorithm has been reported to be almost as good as or even better than iterative algorithms [48, 25] if the clustering threshold parameter is set appropriately. The optimal threshold on our training data was between 0.76 and 0.78 depending on the particular experimental setting. This compares with previously mentioned values and ranges: 0.77 [48] 0.7- 0.9 [25]. We achieved a cluster quality that is comparable with one of the best offline clustering algorithms for text data, namely group-wise average hierarchical clustering.

The single-pass clustering is similar to micro-clustering [3] without the on-demand step. New documents are added to the most similar micro-cluster if the similarity is high enough. If the maximum amount of micro-clusters k is reached inactive clusters are removed. This is similar to our time window. The LSH technique could also be used to speed up micro-clustering.

If we were to execute the on-demand clustering step on a regular basis, several problems would arise within our applications. First of all it would need to done frequently to minimize the delay between the time when the document is ingested into the system and the time when it is available to the user as part of a cluster. Even hourly clustering would mean a significant delay for a news system. Also, it has been reported in [10] that a delay does not necessarily help in new event detection. If each on-demand clustering is done independently, the amount of clusters that are saved to the database is much larger compared to single-pass clustering where one cluster can stretch over a long period of time. Clusters from close-by snapshot times would be very similar creating near duplicates in the database that are hard to detect and filter. A possible solution to this would be the recently proposed evolutionary clustering framework [13] where consecutive clusterings are required to be similar. In this case a previous cluster could be associated with a similar current cluster and saved as a single object in the database. This approach does not, however, support clusters with gaps longer than the clustering interval. Our single-pass clustering supports long cluster lifetimes including gaps up to the duration of the sliding time window. The historical information of the time stamps of documents within a cluster can be easily reconstructed from the database on demand. Finally in [3] there is no mentioning of limiting the number of features which is a potential memory problem.

Our experimental study indicates that utilizing locations, persons, and organizations is of advantage. This is in accordance with previous studies [30, 25]. In [25] one vector for the text and separate vectors for entities and noun phrases are generated. The similarities from comparing the corresponding vectors from different documents are mixed with weights found by regression on a training data set. We integrated the text information and the meta-data into a single vector with emphasized frequencies for meta-data terms. This enables the use of a single hash structure to find a single set of cluster candidates. When using several vectors and similarities the determination of good candidates will be more involved.

When varying the maximum length of cluster centroid vectors we found that the clustering quality on our training increases up to around 1.5k entries and does degrade for larger values. This is in contrast to previous studies that truncated the vectors for efficiency down to 20 [42] or 25 [31] entries. One reason for our observation might be the larger vocabulary and size of our corpus. We recommend to use much higher values.

We did not use any dimensionality reduction techniques like latent semantic indexing (e.g. [9, 18]) because the projection would require additional online computation and we would loose the direct correspondence of feature with words stems that is utilized to generate keywords for each cluster.

9. SUMMARY

We presented a system for high performance online text clustering of a stream of news articles that meets all the identified requirements of the Geospace & Media Tool. The system has been tested on the complete news stream over several weeks and successfully discovered top stories as reported by other news sites. The system will be deployed in the near future to aid members of Congress their staffs in analyzing the daily news in connection with geospatial, census, and human network information.

The textual content of the articles is analyzed and similar articles are grouped into clusters on-the-fly without any assumptions on the number of clusters and without retrieving previous documents. The result is available to the user at any time without additional on-demand clustering steps. The system dynamically adjusts to changing topics by gradually adapting the feature space. Efficiency is ensured by limiting the amount of currently active features and by considering only clusters from a finite time horizon for the assignment of incoming documents. Very large time windows can be supported by using locality-sensitive hashing to summarize the clusters and find the most similar cluster for each document with high probability. We demonstrated the effectiveness and efficiency of the system on the very demanding application of news clustering. The clustering quality is comparable to one of the best non-streaming document clustering algorithms and the architecture can easily support several 10k documents per day on off-the-shelve hardware.

10. ACKNOWLEDGMENTS

We acknowledge the team at Parsons Institute for Information Mapping, The New School, New York, NY, for their collaboration in integrating our text processing methods into the GMT and providing the extracted meta-data for our experiments. We further thank Marc Muntziger for his help in the experiments regarding the meta-data emphasis and Sheik Abdul-Saboor and Tino Hertlein for helping to label the ground truth stories.

11. REFERENCES

[1] C. Aggarwal. Data Streams: Models and Algorithms. Springer, 2007.

[2] C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In Proceedings of the International Conference on Very Large Databases (VLDB), 2003.

[3] C. Aggarwal and P. Yu. A framework for clustering massive text and categorial data. In Proceedings of the SIAM Conference on Data Mining (SDM), 2006.

[4] J. Allan, V. Lavrenko, D. Malin, and R. Swan. Detections, bounds, and timelines: UMass and TDT-3. In Proceedings of the Topic Detection and Tracking Workshop (TDT-3), 2000.

[5] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 37–45, 1998.

[6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In Proceedings of ACM Symposium on Principles of Database Systems (PODS), pages 1–16, 2002.

[7] J. Banerjee and A. Ghosh. Competitive learning mechanisms for scalable, incremental and balanced clustering of streaming texts. In Proceedings of the International Joint Conference on, Neural Networks (IJCNN), volume 4, pages 2697–2702, 2003.

[8] D. Barbara. Requirements for clustering data streams. SIGKDD Explorations, 3(2):23–27, 2002.

[9] M. W. Berry. Survey of Text Mining: Clustering, Classification, and Retrieval. Springer, 2003.

[10] T. Brants and F. Chen. A system for new event detection. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 330–337. ACM Press, 2003.

[11] J. Callan. Document filtering with inference networks. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 262–269. ACM Press, 1996.

[12] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In Proceedings of the SIAM Conference on Data Mining (SDM), 2006.

[13] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In Proceedings of International ACM Conference on Knowledge Discovery and Data Mining (KDD), pages 554–560, 2006.

[14] C. Cieri, D. Graff, M. Liberman, N. Martey, and S. Strassel. The TDT-2 text and speech corpus. In DARPA Broadcast News Workshop, 1999.

[15] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 318–329, 1993.

[16] P. Domingos and G. Hulten. A general method for scaling up machine learning algorithms and its application to clustering. In Proceedings of the International Conference on Machine Learning (ICML), pages 106–113. Morgan Kaufmann, 2001.

[17] P. Domingos and G. Hulten. A general framework for mining massive data streams. Journal of Computational and Graphical Statistics, 12:945–949, 2003.

[18] R. Feldman and J. Sanger. The Text Mining Handbook. Cambridge, 2007.

[19] D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2(2):139–172, 1987.

[20] G. Forman. Tackling concept drift by temporal inductive transfer. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 252–259. ACM Press, 2006.

[21] M. Franz, J. McCarley, T. Ward, and W.-J. Zhu. Unsupervised and supervised clustering for topic tracking. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 310–317, 2001.

[22] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. ACM SIGMOD Record, 34(2):18–26, 2005.

[23] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In Proceedings of the International Conference on Very Large Databases (VLDB), 1999.

[24] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. IEEE Transactions on Knowledge and Data Engineering, 15(3):515–528, 2003.

[25] V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 224–231. ACM Press, 2000.

[26] P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. In Proceedings of the ACM Symposium on Theory of Computing, 1998.

[27] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In Proceedings of the ACM Symposium on Theory of Computing, pages 618–625, 1997.

[28] A. Jain, Z. Zhang, and E. Chang. Adaptive non-linear clustering in data streams. In Proceedings of the ACM International Conference on Information and knowledge management (CIKM), pages 122–131. ACM Press, 2006.

[29] T. Joachims. A statistical learning model of text classification with support vector machines. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 128–136. ACM Press, 2001.

[30] W. Lam, H. Meng, K. Wong, and J. Yen. Using contextual analysis for news event detection. International Journal Of Intelligent Systems, 16(4):525–546, 2001.

[31] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In Proceedings of the Text Mining Workshop at the International ACM Conference on Knowledge Discovery and Data Mining (KDD), pages 16–22. ACM Press, 1999.

[32] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research,, 5:361–397, 2004.

[33] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In Proceedings

of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 37–50, 1992.

[34] D. D. Lewis. Machine learning for text categorization: background and characteristics. In Proceedings of the 21st National Online Meeting, pages 221–226. Information Today, 2000.

[35] S. M., K. G., and K. V. A comparison of document clustering techniques. In Proceedings of the Text Mining Workshop at the International ACM Conference on Knowledge Discovery and Data Mining (KDD), 2000.

[36] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In Proceedings of the International ACM Conference on Knowledge Discovery and Data Mining (KDD), pages 198–207. ACM Press, 2005.

[37] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In Proceedings of IEEE International Conference on Data Engineering, 2002.

[38] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[39] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), pages 357 – 366, 2006.

[40] G. Salton. Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Addison Wesley, 1989.

[41] H. Samet. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufman, 2006.

[42] H. Schuetze and C. Silverstein. Projections for efficient document clustering. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 74–81. ACM Press, 1997.

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

[44] D. Shen, Q. Yang, J.-T. Sun, and Z. Chen. Thread detection in dynamic text message streams. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 35–42. ACM Press, 2006.

[45] A. Smeaton, M. Burnett, F. Crimmins, and G. Quinn. An architecture for efficient document clustering and retrieval on a dynamic collection of newspaper texts. In BCS-IRSG Annual Colloquium on IR Research, Workshops in Computing, 1998.

[46] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. Machine Learning, 23(1):69–101, 1996.

[47] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. IEEE Intelligent Systems, 14(4):32–43, 1999.

[48] Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In Proceedings of the International ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 28–36, 1998.

[49] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In Proceedings of the International ACM Conference on Managemet of Data (SIGMOD), pages 103–114, 1996.

[50] Y.-J. Zhang and Z.-Q. Liu. Self-splitting competitive learning: a new on-line clustering paradigm. IEEE Transactions on Neural Networks, 13(2):369–380, 2002.

[51] Y.-J. Zhang and Z.-Q. Liu. Refining web search engine results using incremental clustering. International Journal of Intelligent Systems, 19(1-2):191–199, 2004.

[52] Y. Zhao and G. Karypis. Hierarchical clustering algorithms for document datasets. Data Mining and Knowledge Discovery, 10(2):141–168, 2005.