Dynamic Pattern Mining: An Incremental Data Clustering Approach

Seokkyung Chung and Dennis McLeod

Department of Computer Science
and Integrated Media System Center
University of Southern California
Los Angeles, California 90089–0781, USA
[*seokkyuc*, *mcleod*]@*usc.edu*

Abstract. We propose a mining framework that supports the identification of useful patterns based on incremental data clustering. Given the popularity of Web news services, we focus our attention on news streams mining. News articles are retrieved from Web news services, and processed by data mining tools to produce useful higher-level knowledge, which is stored in a content description database. Instead of interacting with a Web news service directly, by exploiting the knowledge in the database, an information delivery agent can present an answer in response to a user request. A key challenging issue within news repository management is the high rate of document insertion. To address this problem, we present a sophisticated incremental hierarchical document clustering algorithm using a neighborhood search. The novelty of the proposed algorithm is the ability to identify meaningful patterns (e.g., news events, and news topics) while reducing the amount of computations by maintaining cluster structure incrementally. In addition, to overcome the lack of topical relations in conceptual ontologies, we propose a topic ontology learning framework that utilizes the obtained document hierarchy. Experimental results demonstrate that the proposed clustering algorithm produces high-quality clusters, and a topic ontology provides interpretations of news topics at different levels of abstraction.

1 Introduction

With the rapid growth of the World Wide Web, Internet users are now experiencing overwhelming quantities of online information. Since manually analyzing the data becomes nearly impossible, the analysis would be performed by automatic data mining techniques to fulfill users' information needs quickly.

On most Web pages, vast amounts of useful knowledge are embedded into text. Given such large sizes of text datasets, mining tools, which organize the text datasets into structured knowledge, would enhance efficient document access. This facilitates information search and at the same time, provides an efficient framework for document repository management as the number of documents becomes extremely huge.

Given that the Web has become a vehicle for the distribution of information, many news organizations are providing newswire services through the Internet.

Given this popularity of the Web news services, we have focused our attention on mining patterns from news streams.[1]

The simplest document access method within Web news services is keyword-based retrieval. Although this method seems effective, there exist at least three drawbacks. First, if a user chooses irrelevant keywords (due to broad and vague information needs or unfamiliarity with the domain of interest), retrieval accuracy will be degraded. Second, since keyword-based retrieval relies on the syntactic properties of information (e.g., keyword counting),[2] *semantic gap* cannot be overcome. Third, only expected information can be retrieved since the specified keywords are generated from users' knowledge space. Thus, if users are unaware of the airplane crash that occurred yesterday, then they cannot issue a query about that accident even though they might be interested.

The first two drawbacks stated above have been addressed by query expansion based on domain-independent ontologies [47]. However, it is well known that this approach leads to a degradation of precision. That is, given that the terms introduced by term expansion may have more than one meaning, using additional terms can improve recall, but decrease precision. Exploiting a manually developed ontology with a controlled vocabulary is helpful in this situation [27, 28, 29]. However, although ontology-authoring tools have been developed in the past decades, manually constructing ontologies whenever new domains are encountered is an error-prone and time-consuming process. Therefore, integration of knowledge acquisition with data mining, which is referred to as *ontology learning*, becomes a must [32].

In this paper, we propose a mining framework that supports the identification of meaningful patterns (e.g., topical relations, topics, and events that are instances of topics) from news stream data. To build a novel framework for an intelligent news database management and navigation scheme, we utilize techniques in information retrieval, data mining, machine learning, and natural language processing.

To facilitate information navigation and search on a news database, we first identify three key problems.

1. *Vague information needs.* Sometimes, defining keywords for a search is not an easy task, especially when a user has vague information needs. Thus, a reasonable starting point would be provided to assist the user.
2. *Lack of topical relations in concept-based ontologies.* In order to achieve rich semantic information retrieval, an ontology-based approach would be provided. However, as discussed in Agirre *et al.* [2], one of the main problems with concept-based ontologies is that topically related concepts and terms

[1] In this paper, we are concerned with (news) articles, which are also referred to as documents.

[2] Like Latent Semantic Indexing (LSI) [8], the vector space model based on keyword counting can be augmented with semantics by combining other methods (e.g., Singular Value Decomposition). However, keyword-based retrieval in this paper is referred to as the method relying on only simple keyword counting.

are not explicitly linked.[3] That is, there is no relation between *court-attorney*, *kidnap-police*, etc. Thus, concept-based ontologies have a limitation in supporting a topical search. For example, consider the Sports domain ontology that we have developed in our previous work [27, 28, 29]. In this ontology, "Kobe Bryant", who is an NBA basketball player, is related with terms/concepts in Sports domain. However, for the purpose of query expansion, "Kobe Bryant" needs to be connected with a "court trial" concept if a user keeps "Kobe Bryant court trial" in mind. Therefore, it is essential to provide explicit links between topically related concepts/terms.

3. *High rate of document insertion.* As several hundred news articles are published everyday at a single Web news site, triggering the whole mining process whenever a document is inserted to the database is computationally impractical. To cope with such a dynamic environment, efficient incremental data mining tools need to be developed.

The first of the three problems can be approached using clustering. A collection of documents is easy to skim if similar articles are grouped together. If the news articles are hierarchically classified according to their topics, then a query can be formulated while a user navigates a cluster hierarchy. Moreover, clustering can be used to identify and deal with near-duplicate articles. That is, when news feeds repeat stories with minor changes from hour to hour, presenting only the most recent articles is probably sufficient.

To remedy the second problem, we present a *topic ontology*, which is defined as a collection of concepts and relations. In a topic ontology, concept is defined as a set of terms that characterize a topic. We define two generic kinds of relations, *generalization* and *specialization*. The former can be used when a query is generalized to increase recall or broaden the search. On the other hand, the latter is useful when refining the query. For example, when a user is interested in someone's court trial but cannot remember the name of a person, then specialization can be used to narrow down the search.

To address the third problem, we propose a sophisticated incremental hierarchical document clustering algorithm using a neighborhood search. The novelty of the proposed algorithm is the ability to identify news event clusters as well as news topic clusters while reduce the amount of computation by maintaining cluster structure incrementally. Learning topic ontologies can be performed on the obtained document hierarchy.

Figure 1 illustrates the main parts of the proposed framework. In the information gathering stage, a Web crawler retrieves a set of news documents from a news Web site (e.g., CNN). Developing an intelligent Web crawler is another research area, and it is not our main focus. Hence, we implement a simple Web spider, which downloads news articles from a news Web site on a daily basis. The retrieved documents are processed by data mining tools to produce useful

[3] Although there exist different types of term association relationships in WordNet [36] such as "Bush versus President of US" as synonym, or "G.W. Bush versus R. Reagan" as coordinate terms, these types of relationships are limited to addressing topical relationships.

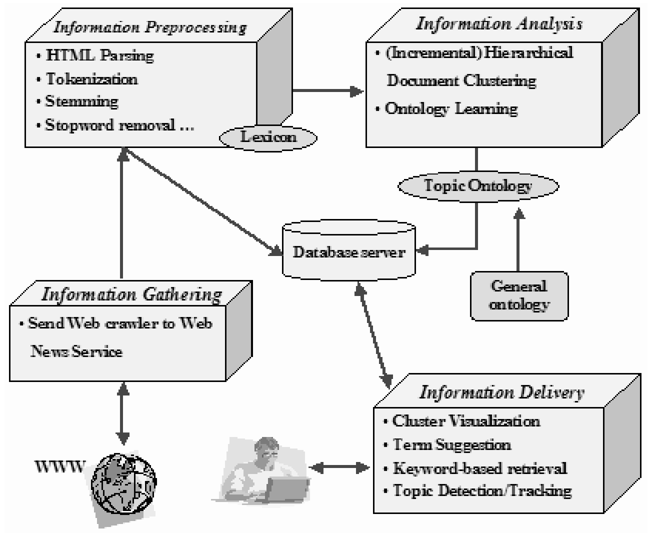


Fig. 1. Overview of a proposed framework

higher-level knowledge (e.g., a document hierarchy, a topic ontology, etc), which is stored in a content description database. Instead of interacting with a Web news service directly, by exploiting knowledge in the database, an information delivery agent can present an answer in response to a user request.

Main contributions of our work are twofold. First, despite the huge body of research efforts on document clustering [33, 30, 22, 31, 52], little work has been conducted in the context of incremental hierarchical news document clustering. To address the problem of frequent document insertions into a database, we have developed an incremental hierarchical clustering algorithm using a neighborhood search. Since the algorithm produces a document cluster hierarchy, it can identify event level clusters as well as topic level clusters. Second, to address the lack of topical relations in concept-based ontologies, we propose a topic ontology learning framework, which can interpret news topics at multiple levels of abstraction.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 discusses the information preprocessing step. In Section 4, we explain the information analysis component, which is a key focus of this paper. Section 5 presents experimental results. Finally, we conclude the paper and provide our future plans in Section 6.

2 Related Work

The most relevant research areas to our work are Topic Detection and Tracking (TDT) and document clustering. Section 2.1 presents a brief overview on TDT work. In Section 2.2, a survey on previous document clustering work is provided.

Finally, Section 2.3 introduces previous work on intelligent news services, which utilize document clustering and TDT.

2.1 Topic Detection and Tracking

Over the past six years, the information retrieval community has developed a new research area, called Topic Detection and Tracking (TDT) [4, 5, 10, 48, 49]. The main goal of TDT is to detect the occurrence of a novel event in a stream of news stories, and to track the known event. In particular, there are three major components in TDT.

1. *Story segmentation.* It segments a news stream (e.g., including transcribed speech) into topically cohesive stories. Since online Web news (in HTML format) is supplied in segmented form, this task only applies to audio or TV news.
2. *First Story Detection (FSD).* It identifies whether a new document belongs to an existing topic or a new topic.
3. *Topic tracking.* It tracks events of interest based on sample news stories. It associates incoming news stories with the related stories, which were already discussed before. It can be also asked to monitor the news stream for further stories on the same topic.

In Allan *et al.* [4], the notion of *event* is first defined. *Event* is defined as "some unique thing that happens at some point in time". Hence, an event is different from a topic. For example, "airplane crash" is a topic while "Chinese airplane crash in Korea in April 2002" is an event. Thus, there exists M-1 mapping between event and topic (i.e., multiple events can be on a same topic). Note that it is important to identify events as well as topics. Although the user may not be interested in a flood topic, in general, she may be interested in documents about a flood event in her home town. Thus, a news recommendation system must be able to distinguish different events within a same topic.

Yang *et al.* introduced an important property of news events, referred to as *temporal locality* [48]. That is, news articles discussing the same event tend to be temporally proximate. In addition, most of the events (e.g., flood, earthquake, wildfire, kidnapping) have short duration (e.g., 1 week - 1 month). They exploited these heuristics when computing similarity between two news articles.

The most popular method in TDT is to use a simple incremental clustering algorithm, which is shown in Figure 2. Our work starts by addressing the limitations of this algorithm.

2.2 Document Clustering

In this section, we classify the widely used document clustering algorithms into two categories (partition-based clustering and hierarchical clustering), and provide a concise overview for each of them.

1. Initially, only one news article is available, and it forms a singleton cluster.
2. For an incoming document (d_*), we compute the similarity between d_* and pre-generated clusters. The similarity is computed by the distance between d_* and the representative of the cluster.
3. Selects the cluster (C_i) that has the maximum proximity with d_*.
4. If the similarity between d_* and C_i exceeds the pre-defined threshold, then all documents in C_i are considered as related stories to d_* (topic tracking), and d_* is assigned to C_i.
   Otherwise, d_* is considered as a novel story (first story detection), and a new cluster for d_* is created.
5. Repeat 2-4 whenever a new document appears in a stream.

Fig. 2. The incremental document clustering algorithm in TDT

Partition-based Clustering Partition-based clustering decomposes a collection of documents, which is optimal with respect to some pre-defined function. Typical methods in this category include center-based clustering, Gaussian Mixture Model, etc.

Center-based algorithms identify the clusters by partitioning the entire dataset into a pre-determined number of clusters (e.g., K-means clustering), or an automatically derived number of clusters (e.g., X-means clustering) [9, 23, 13, 16, 30, 37, 39].

The most popular and the best understood clustering algorithm is K-means clustering [13]. The K-means algorithm is a simple but powerful iterative clustering method to partition a dataset into K disjoint clusters, where K must be determined beforehand. The idea of the algorithm is to assign points to the cluster such that the sum of the mean square distance of points to the center of the assigned cluster is minimized.

While the K-means clustering approach works in a metric space, medoid-based method works with a similarity space [23, 37]. It uses the medoids (representative sample objects) instead of the means (e.g., the centers of clusters) such that the sum of the distances of points to their closest medoid is minimized.

Although the center-based clustering algorithms have been widely used in document clustering, there exist at least five serious drawbacks. First, in many center-based clustering algorithms, the number of clusters (K) needs to be determined beforehand. Second, the algorithm is sensitive to an initial seed selection.

Depending on the initial points, it is susceptible to a local optimum. Third, it can model only a spherical (K-means) or ellipsoidal (K-medoid) shape of clusters. Thus, the non-convex shape of clusters cannot be modeled in center-based clustering. Forth, it is sensitive to outliers since a small amount of outliers can substantially influence the mean value. Finally, due to the nature of iterative scheme in producing clustering results, it is not relevant for incremental datasets.

Hierarchical Agglomerative Clustering Hierarchical (agglomerative) clustering (HAC) finds the clusters by initially assigning each document to its own cluster and then repeatedly merging pairs of clusters until a certain stopping condition is met [13, 18, 26, 19, 52]. Consequently, its result is in the form of a tree, which is referred to as a *dendrogram.* A dendrogram is represented as a tree with numeric levels associated to its branches.

The main advantage of HAC lies in its ability to provide a view of data at multiple levels of abstraction. However, since HAC builds a dendrogram, a user must determine where to cut the dendrogram to produce actual clusters. This step is usually done by human visual inspection, which is a time-consuming and subjective process. Moreover, the computational complexity of HAC is more expensive than that of partition-based clustering. In partition-based clustering, the computational complexity is $\mathrm{O}(nKI)$ where n is the number of documents, K is the number of clusters, and I is the number of iterations, respectively. In contrast, HAC takes $O(n^3)$ if pairwise similarities between clusters are changed when two clusters are merged. However, the complexity can be reduced to $O(n^2logn)$ if we utilize a priority queue [52].

2.3 Intelligent News Services Systems

The one of the most successful intelligent news services is NewsBlaster [34]. The basic idea of NewsBlaster is to group the articles on the same story using clustering, and present one story using multi-document summarization. Thus, the main goal of NewsBlaster is similar to ours in that both aim to propose intelligent news analysis/delivery tools. However, the underlying methodology is different. For example, with respect to clustering, NewsBlaster is based on the clustering algorithm in Hatzivassiloglou *et al.* [22]. Main contributions of their work is to augment document representation using linguistic features. However, rather than developing their own clustering algorithm, they used conventional HAC, which has the drawbacks as discussed in Section 2.2.

Recent attempts present other intelligent news services like NewsInEssence [41, 42], or QCS (Query, Cluster, Summarize) [14]. Both services utilize a similar approach to NewsBlaster in that they separate the retrieved documents into topic clusters, and create a single summary for each topic cluster. However, their main focus does not lie in developing a novel clustering algorithm. For example, QCS utilizes generalized spherical K-means clustering whose limitations have been addressed in Section 2.2.

Therefore, it is worthwhile to develop a sophisticated document clustering algorithm that can overcome the drawbacks of previous document clustering work. In particular, the developed algorithm must address the special requirements in news clustering such as high rate of document insertion, or ability to identify event level clusters as well as topic level clusters.

3 Information Preprocessing

The information preprocessing step extracts meaningful information from unstructured text data and transforms it into structured knowledge. As shown in Figure 1, this step is composed of the following standard IR tools.

- *HTML preprocessing*. Since downloaded news articles are in HTML format, we remove irrelevant HTML tags for each article and extract meaningful information.
- *Tokenization*. Its main task is to identify the boundaries of the terms.
- *Stemming*. There can be different forms for the same terms (e.g., *students* and *student*, *go* and *went*). These different forms of the same term need to be converted to their roots. Toward this end, instead of solely relying on Porter stemmer [40], in order to deal with irregular plural/tense, we combine Porter stemmer with the lexical database [35].
- *Stopwords removal*. Stopwords are the terms that occur frequently in the text but do not carry useful information. For example, *have*, *did*, and *get* are not meaningful. Removing such stopwords provide us with a dimensionality reduction effect. We employ the stopword list that was used in Smart project [44].

After preprocessing, a document is represented as a vector in an n-dimensional vector space [44]. The simple way to do this is to employ the Bag-Of-Word (BOW) approach. That is, all content-bearing terms in the document are kept and any structure of text or the term sequence is ignored. Thus, each term is treated as a feature and each document is represented as a vector of certain weighted term frequencies in this feature space.

There are several ways to determine the weight of a term in a document. However, most methods are based on the following two heuristics.

- Important terms occur more frequently within a document than unimportant terms do.
- The more times a term occurs throughout all documents, the weaker its discriminating power becomes.

The term frequency (TF) is based on the first heuristic. In addition, TF can be normalized to reflect different document lengths. Let $freq_{ij}$ be the number of t_i's occurrence in a document j, and l_j be the length of the document j. Then, term frequency (tf_{ij}) of t_i in the document j is defined as follows:

$$tf_{ij} = \frac{freq_{ij}}{l_j} \tag{1}$$

	kidnap	abduct	child	boy	police	search	missing	investigate	suspect	return	home
d_1	1	0	1	0	1	1	0	1	0	0	0
d_2	1	1	1	1	1	0	1	1	1	0	0
d_3	0	1	0	1	0	0	1	0	0	1	1

Table 1. A sample illustrative example for document×term matrix. For simplicity, each document vector is represented as boolean values instead of TF-IDF values

The document frequency (DF) of the term (the percentage of the documents that contain this term) is based on the second heuristic. A combination of TF and DF introduces TF-IDF ranking scheme, which is defined as follows:

$$w_{ij} = tf_{ij} \times log \frac{n}{n_i} \tag{2}$$

where w_{ij} is the weight of t_i in a document j, n is the total number of documents in the collection, and n_i is the number of documents where t_i occurs at least once.

The above ranking scheme is referred to as static TF-IDF since it is based on static document collection. However, since documents are inserted incrementally, IDF values are initialized using a sufficient amount of documents (i.e., the document frequency is generated from training corpus). After then, IDF is incrementally updated as subsequent documents are processed. In particular, we employ an incremental update of IDF value proposed by Yang *et al.* [48].

Finally, to measure closeness between two documents, we use the Cosine metric, which measures the similarity of two vectors according to the angle between them [44]. Thus, vectors pointing to similar directions are considered as representing similar concepts. The cosine of the angles between two m-dimensional vectors (x and y) is defined by

$$Similarity(x, y) = Cosine(x, y) = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{||x||_2 \cdot ||y||_2} \tag{3}$$

4 Information Analysis

This section presents the information analysis component of Figure 1. Section 4.1 illustrates a motivating example for the proposed incremental clustering algorithm. In Section 4.2, a non-hierarchical incremental document clustering algorithm using a neighborhood search is presented. Section 4.3 explains how to extend the algorithm into a hierarchical version. Finally, Section 4.4 shows how to build a topic ontology based on the obtained document hierarchy.

Notation	Meaning
n	The total number of documents in a database
d_*	A new document
d_i	An i-th document
ϵ	Threshold for determining the neighborhood
$N_\epsilon(d_i)$	ϵ-neighborhood for d_i
D_{d_i}	The set of documents that contain any term of d_i
C_{d_i}	The set of clusters that contain any neighbor of d_i
$\|A\|$	The size of a set A where A can be a neighborhood or cluster
df_{ij}	Document frequency of a term t_i within a set A_j
w_{ij}	TF-IDF value for a term t_i for a document d_j
S_j	Signature vector for a set A_j
s_i^j	$i-$th component of S_j

Table 2. Notations for incremental non-hierarchical document clustering

4.1 A Motivating Example

To illustrate a simple example, consider the following three documents (whose document×term matrix is shown in Table 1).

1. d_1: A child is kidnapped so police starts searching.
2. d_2: Police found the suspect of child kidnapping.
3. d_3: An abducted boy safely returned home.

In the above three documents, although d_1 and d_2 are similar, and d_2 and d_3 are similar, d_1 and d_3 are completely dissimilar since they share no terms. Consequently, transitivity relation does not hold. Why does this happen? We provide explanations to this question in terms of three different perspectives.

1. *Fuzzy similarity relation.* As discussed in the fuzzy theory [50], the similarity relation does not satisfy transitivity. To make it satisfy transitivity, a fuzzy transitivity closure approach was introduced. However, this approach is not scalable with the number of data points.
2. *Inherent characteristic of news.* As discussed in Allan *et al.* [4], event is considered as an evolving object through some time interval (i.e., content of news articles on the same story are changed throughout time). Hence, although the documents belong to a same event, the terms the documents use would be different if they discuss different aspects of the event.
3. *Language semantics.* The diverse term usage for a same meaning (e.g., kidnap and abduct) needs to be considered. Using only a syntactic property (e.g., keyword counting) aggravates the problem.

The transitivity is related with document insertion order in incremental clustering. Consider the TDT incremental clustering algorithm in Figure 2. If the order of document insertion is "$d_1d_2d_3$", then one cluster ($\{\{d_1, d_2, d_3\}\}$) is obtained. However, if the order is "$d_1d_3d_2$", then two clusters ($\{\{d_1, d_2\}, \{d_3\}\}$) are obtained. Although the order of document insertion is fixed (because the document is inserted whenever it is published), it is undesirable if the clustering result significantly depends on the insertion order. Regardless of the input order, the successful algorithm should produce a single cluster, $\{\{d_1, d_2, d_3\}\}$.

4.2 A Proposed Incremental Non-hierarchical Document Clustering Algorithm using a Neighborhood Search

Before we present detailed discussions on the proposed clustering algorithm, definitions for basic terminology are provided first. In addition, Table 2 shows the notations, which will be used throughout this paper.

Definition 1 similar. If $Similarity(d_i, d_j) \geq \epsilon$, then a document d_i is referred to as similar to a document d_j.

Definition 2 $N_\epsilon(d_i)$. ϵ-neighborhood for d_i is $\{x : Similarity(x, d_i) \geq \epsilon\}$.

That is, ϵ-neighborhood for a document d_i is defined as a set of documents, which are similar to d_i. In this paper, ϵ-neighborhood and neighborhood are used interchangeably.

Definition 3 neighbor. A document d_j is defined as a neighbor of d_i if and only if $d_j \in N_\epsilon(d_i)$.

The proposed clustering algorithm is based on the observation that a property of an object would be influenced by the attributes of its neighbors. Examples of such attributes are the properties of the neighbors, or the percentage of neighbors that fulfill a certain constraint. The above idea can be translated into clustering perspective as follows: a cluster label of an object depends on the cluster labels of its neighbors.

Recent data mining research has proposed density-based clustering such as Shared Nearest Neighbors (SNN) clustering [15, 24]. In SNN, the similarity between two objects is defined as the number of k-nearest neighbors they share. Thus, the basic motivation of SNN clustering is similar to ours, however, as we will explain in Section 4.3, the detailed approach is completely different.

Figure 3 shows the proposed incremental clustering algorithm. Initially, we assume that only one document is available. Thus, this document itself forms a singleton cluster. Adding a new document to existing cluster structure proceeds in three phases: neighborhood search, identification of an appropriate cluster for a new document, and re-clustering based on local information. In what follows, these three steps are explained in detail.

```
Step 1. Initialization:
    Document $d_0$ forms a singleton cluster $C_0$.

Step 2. Neighborhood search:
    Given a new incoming document $d_*$, obtain $N_\epsilon(d_*)$ by performing
    a neighborhood search.

Step 3. Identification of a cluster that can host a new document:
    Compute the similarity between $d_*$ and a cluster $C_i \in C_{d_*}$.

    Based on the value obtained from above,
    if there exists a cluster ($C_j$) that can host $d_*$, then
        add $d_*$ to the cluster and update the $DCF_j$.
    Otherwise,
        create a new cluster for $d_*$ and
        create a corresponding DCF vector for this new cluster.

Step 4. Re-clustering:
    Let $C_j$ be the cluster that hosts $d_*$.
    If $C_j$ is not a singleton cluster, then trigger merge operation.

Step 5.
    Repeat Step 2-4 whenever a new document appears in a stream.
```

Fig. 3. The incremental non-hierarchical document clustering algorithm

Neighborhood search Achieving an efficient neighborhood search is important in the proposed clustering algorithm. Since we deal with documents in this research, we can rely on an inverted index for the purpose of the neighborhood search.[4] In an inverted index [44], the index associates a set of documents with terms. That is, for each term t_i, we build a document list that contains all documents containing t_i. Given that a document d_i is composed of t_1, ... ,t_k, to identify similar documents to d_i, instead of checking whole document dataset, it is sufficient to examine the documents that contain any t_i. Thus, given a document d_i, identifying the neighborhood can be accomplished in $O(|D_{d_i}|)$.

[4] Note that the neighborhood search can be supported with Multi-Dimensional Index (MDI) structure [6, 20, 7] coupling with dimensionality reduction (e.g., wavelet transforms [11] or Fourier transforms [3]) if the proposed algorithm is extended into other data types such as time-series.

Identification of an appropriate cluster To assign an incoming document (d_*) to the existing cluster, the cluster, which can host d_*, needs to be identified using the neighborhood of d_*. If there exists such a cluster, then d_* is assigned to the cluster. Otherwise, d_* is identified as an outlier and forms a singleton cluster.

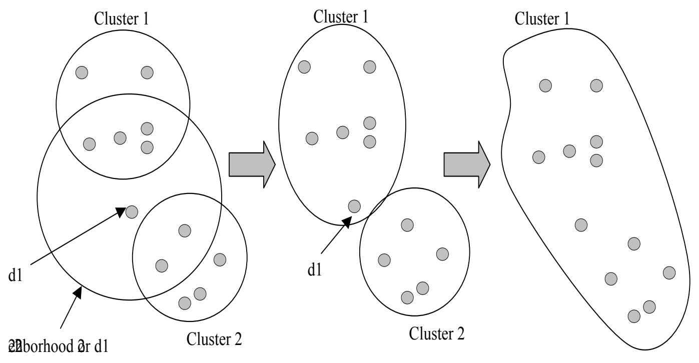
Toward this end, the set of candidate clusters (C_{d_*}) is identified by selecting the cluster that contains any document belonging to $N_\epsilon(d_*)$. Subsequently, the cluster, which can host a new document, is identified by using one of the following three methods.

1. *Considering the size of an overlapped region.* Select the cluster that has the largest number of its members in $N_\epsilon(d_*)$. This approach only considers the number of documents in the overlapped region, and ignores the proximity between neighbors and d_*.
2. *Exploiting weighted voting.* The similarities between each neighbor of d_* and the candidate clusters are measured. Then, the similarity values are aggregated using weighted voting. That is, the weight is determined by the similarity between the proximity of a neighbor to the new document. Thus, each neighbor can vote for its class with a weight proportional to its proximity to the new document.
   Let W_j be a weight for representing the proximity of n_j to the new document (e.g., Cosine similarity between n_j and the new document). Then, the most relevant cluster (C_*) is selected based on the following formula:
   $$C_* = argmax_{C_k \in C_{d_*}} \sum_{n_j \in N_\epsilon(d_*)} W_j \cdot Similarity(n_j, S_k) \tag{4}$$
   Equation (4) mitigates the problem of the previous method by considering the weight W_j. Moreover, it still favors the cluster with a large size of overlapped region to $N_\epsilon(d_*)$ by summing up the weighted similarity.
3. *Exploiting a signature vector.* While the weighted voting approach is effective, it is computationally inefficient since the similarities between all neighbors and all candidate clusters need to be computed. Instead, we employ a simple but effective approach, which measures the similarity between the signature vector of the neighborhood and that of the candidate clusters.

The signature vector should be composed of terms that reflect the main characteristics of the documents within a set. For example, the center of a cluster would be a signature vector for the cluster. For each term t_i in the set A_j (e.g., cluster/neighborhood), we compute the weight for the signature vector using the following formula:

$$s_i^j = \frac{df_{ij}}{|A_j|} \cdot \frac{\sum_{d_k \in A_j} w_{ik}}{|A_j|} \tag{5}$$

In Equation (5), the first factor measures the normalized document frequency



Step 1: Check whether d1 can be added to cluster 1

Step 2[illegible]d d1 to cluster 1

Step 2[illegible] cluster 1 and cluster [illegible]he[illegible]at[illegible]he [illegible] constra[illegible]t

Fig. 4. Illustration of a re-clustering phase

within a set, and the second factor measures the sum of the weight for the term over the whole documents within a set.

Next, the notion of Document Cluster Feature (DCF) vector[5] is presented as follows:

Definition 4 DCF. Document Cluster Feature (DCF) vector for a cluster C_i is defined as a triple $DCF_i = (N_i, DF_i, W_i)$ where N_i is the number of documents in C_i, DF_i is a document frequency vector for C_i, and W_i is a weight sum vector for C_i, respectively.

Theorem 5 Additivity of DCF. *Let $DCF_i = (N_i, DF_i, W_i)$ and $DCF_j = (N_j, DF_j, W_j)$ be the document cluster feature vectors for C_i and C_j, respectively. Then, DCF for a new cluster (by merging C_i and C_j) is defined by $(N_i + N_j, DF_i + DF_j, W_i + W_j)$.*

Proof. It is straightforward by simple linear algebra.

To compute the similarity between a document and a cluster, we only need signature vectors of the cluster and the document. However, the signature vector does not need to be recomputed as a new document is inserted to the cluster. This property is based on the additivity of DCF. Since S_i (a signature vector for C_i) can be directly reconstructed from DCF_i, instead of recomputing S_i whenever a new document is inserted into C_i, the DCF_i only needs to be updated using the additivity of DCF.

[5] The basic notion of DCF is motivated by Cluster Feature (CF) in BIRCH clustering [51].

Notation	Meaning
ST_{C_i}	A collection of specific terms for C_i
T	Virtual time
$df^i(T)$	Document frequency of a term t_i in whole documents at time T
$df^{ij}_{IN}(T)$	Document frequency of a term t_i within C_j at time T
K	Number of clusters at level 1 at time T
$df^{ij}_{OUT}(T)$	A quantitative value representing how much t_i occurs outside the cluster C_j at time T
$Sel^i_j(T)$	Selectivity of a term t_i for the cluster C_j at time T
C^j_i	A i-th cluster at level j

Table 3. Notations for incremental hierarchical document clustering

In sum, if there exists a cluster (C_i) that can host a new document, then the new document is assigned to C_i and the DCF_i is updated. Otherwise, a new cluster for d_* and a DCF vector for this cluster are created.

Re-clustering If d_* is assigned to C_i, then a merge operation needs to be triggered. This is based on a locality assumption [43]. Instead of re-clustering the whole dataset, we only need to focus on the clusters that are affected by the new document. That is, a new document is placed in the cluster, and a sequence of cluster re-structuring processes is performed only in regions that have been affected by the new document. Figure 4 illustrates this idea. As shown, clusters that contain any document belonging to the neighborhood of a new document need to be considered.

4.3 How to Extend the Non-hierarchical Clustering Algorithm into a Hierarchical Version?

When the algorithm in Figure 3 is applied to a news article dataset, different event clusters[6] can be obtained. Since our goal is to generate a cluster hierarchy, all event clusters on the same topic need to be combined together. For example, to reflect a court trial topic, all court trial event clusters at level 1 should be merged in a single cluster at level 2. However, in many cases, this becomes a difficult task due to the extremely high term-frequency of named entities within a document. Named entities are people/organization, time/date and location, which play a key role in defining "who", "when", and "where" of a news event.

[6] These event clusters are defined at level 1. Note that level 0 corresponds to the lowest level in a cluster tree (i.e., each document itself forms a singleton cluster at level 0). Thus, clusters at level 1 are expected to contain similar documents on a certain event (i.e., event clusters) while clusters at level 2 are expected to contain similar documents on a certain topic (i.e., topic clusters).

Thus, although two different event clusters belong to the same topic, similarity between the clusters becomes extremely low, consequently, the task of merging different event clusters (on a same topic) is not simple.

To address the above problem, we illustrate how to extend the algorithm (in Figure 3) into a hierarchical version. Table 3 summarizes the notations that will be used in this section. Before presenting a detailed discussion, necessary terminology is first defined.

Definition 6. Specific term (ST). A specific term for a cluster C_i is a term, which frequently occurs within a cluster C_i, but rarely occurs outside of C_i. The collection of specific terms for C_i is denoted by ST_{C_i}.

Definition 7. Virtual time (T). Virtual time T is initialized by 0. At any time T, only one operation (e.g., document insertion and cluster merge) can be performed. In addition, T is increased by one only when an operation is performed.

Let $df^i(T)$ be the document frequency of a term t_i in whole document dataset at time T. Then, the document frequency of t_i at time $T+1$ is defined as follows:

$$df^i(T+1) = \begin{cases} df^i(T)+1, & \text{if } d_* \text{ is inserted at } T \text{ and } d_* \text{ contains } t_i, \\ df^i(T), & \text{Otherwise} \end{cases} \tag{6}$$

Let $df^{ij}_{IN}(T)$ be the document frequency of a term t_i within a C_j at time T. Then $df^{ij}_{IN}(T+1)$ is recursively defined as follows:

$$df^{ij}_{IN}(T+1) = \begin{cases} df^{ij}_{IN}(T)+1, & \text{if } d_* \text{ is inserted to } C_j \text{ at } T \text{ and } d_* \text{ contains } t_i \\ df^{ij}_{IN}(T), & \text{Otherwise} \end{cases} \tag{7}$$

We denote $K(T)$ as a number of clusters at level 1 at time T. Then, $K(T+1)$ is defined as follows:

$$K(T+1) = \begin{cases} K(T), & \text{if } d_* \text{ is inserted to an existing cluster at } T \\ K(T)+1, & \text{if } d_* \text{ itself forms a new cluster at } T \\ K(T)-1, & \text{if two clusters are merged at } T \end{cases} \tag{8}$$

Although $df^i(T+1) - df^{ij}_{IN}$ could be considered for representing how much t_i occurs outside C_j at $T+1$, it is not sufficient if our goal is to quantify how much t_i is informative for C_j. This is because the number of clusters can also affect on how much t_i discriminates C_j from other clusters. Thus, $df^{ij}_{OUT}(T+1)$, which represents how much t_i occurs outside C_j at time $T+1$, can be defined as follows:

$$df^{ij}_{OUT}(T+1) = \frac{df^i(T+1) - df^{ij}_{IN}(T+1)}{K(T+1)-1} \tag{9}$$

Finally, the selectivity of a term t_i for the cluster C_j at time $T+1$ is defined as follows:

$$Sel_j^i(T+1) = log\frac{df_{IN}^{ij}(T+1)}{df_{OUT}^{ij}(T+1)} \tag{10}$$

In sum, Equation (10) assigns more weight to the terms occurring frequently within C_j, and occurring rarely outside of C_j. Therefore, a term with high selectivity for C_i can be a candidate for ST_{C_i} .

Based on the definition of ST, the proposed hierarchical clustering algorithm is described. While clusters at level 1 are generated using the algorithm in Figure 3, if no more documents are inserted to a certain cluster at level 1 during the pre-defined time interval, then we assume that the event for the cluster ends,[7] and associate ST with this cluster at level 1. We then perform a neighborhood search for this cluster at level 2. Since ST reflects the most specific characteristics for the cluster, it is not helpful if two topically similar clusters (but different events) need to be merged. Hence, when we build a vector for C_i^j, terms in ST (for C_i^j) are not included for building a cluster vector.

At this moment, it is worthwhile to compare our algorithm with the SNN approach [24, 15]. The basic strategy of SNN clustering is as follows: It first constructs the nearest neighbor graph from the sparsified similarity matrix, which is obtained by keeping only k-nearest neighbor of each entry. Next, it identifies representative points by choosing the points that have high density, and removes noise points that have low density. Finally, it takes connected components of points to form clusters.

The key difference between SNN and our approach is that SNN is defined on static datasets while ours can deal with incremental datasets. The re-clustering phase, and special data structures (e.g., DCF or signature vector) make our algorithm more suitable for incremental clustering than SNN. The second distinction is how a neighborhood is defined. In SNN, a neighborhood is defined as a set of k-nearest neighbors while we use ϵ-neighborhood. Thus, as discussed in Han *et al.* [21], the neighborhood constructed from k-nearest neighbors is local in that the neighborhood is defined narrowly in dense regions while it is defined more widely in sparse regions. However, for document clustering, a global neighborhood approach produces more meaningful clusters. The third distinction is that we intend to build a cluster hierarchy incrementally. In contrast, SNN does not focus on hierarchical clustering. Finally, our algorithm can easily identify singleton clusters. This is especially important in our application since an outlier document on a in a news stream may imply a valuable fact (e.g., a new event or technology that has not been mentioned in previous articles). In contrast, SNN overlooks the importance of singleton clusters.

[7] This assumption is based on the temporal proximity of an event [48].

Event	Specific features
Court trial 1	winona, ryder, *actress, shoplift*, beverly
Court trial 2	andrea, yates, *drown, insanity*
Court trial 3	blake, bakley, *actor*
Court trial 4	moxley, martha, kennedy, michael
Kidnapping 1	elizabeth, smart, utah, salt, lake
Kidnapping 2	jessica, holly, soham, cambridgeshire, england
Kidnapping 3	weaver, ashlei, *miranda*, gaddis
Kidnapping 4	avila, samantha, runnion
Earthquake 1	san, giuliano, puglia, italy, sicily, etna
Earthquake 2	china, bachu, beijing, xinjiang
Earthquake 3	algeria, algerian
Earthquake 4	iran, qazvin

Table 4. A sample specific terms for the clusters at level 1. The term with regular font denotes NE. Thus, this supports the argument that NE plays a key role in defining specific details of events

Topic	Specific features
Court trial	attorney court defense evidence jury kill law legal murder prosecutor testify trial
Kidnapping	abduct disappear enforce family girl kidnap miss parent police
Earthquake	body collapse damage earthquake fault hit injury magnitude quake victim
Airplane crash	accident air aircraft airline aviate boeing collision crash dead flight passenger pilot safety traffic warn

Table 5. A sample specific terms for the clusters at level 2

4.4 Building a Topic Ontology

A topic ontology is a collection of concepts and relations. One view of a concept is as a set of terms that characterize a topic. We define two generic kinds of relations, specialization and generalization. The former is useful when refining a query while the latter can be used when generalizing a query to increase recall or broaden the search.

Table 4 and Table 5 illustrate the sample specific terms for the selected events/topics. As shown, with respect to the news event, we observed that the specific details are captured by the lower levels (e.g., level 1), while higher levels (e.g., level 2) are abstract. We can also generate general terms for the node, which is defined as follows:

Event	General features
Court trial 1	arm arrest camera count delay drug hill injury order store stand target victim

Table 6. General terms for the court trial cluster 1 in Table 4

Definition 8. General term (GT). A general term for a cluster C_i is a term, which frequently occurs within a cluster C_i, and also frequently occurs outside of C_i. A collection of general terms for C_i is denoted by GT_{C_i}.

Thus, in comparison with ST, the selectivity of GT is less than that of ST. Those ST and GT constitute the concepts of a topic ontology.[8]

Table 6 shows GT for the "court trial 1" cluster in Table 4. When the "Winona Ryder court trial" cluster (C_1) is considered, ST_{C_1} represents the most specific information for "Winona Ryder court trial event", GT_{C_1} carries the next most specific information for the event, and specific terms for the court trial cluster describe the general information for the event. Therefore, we can conclude that a topic ontology can characterize a news topic at multiple levels of abstraction.

Human-understandable information needs to be associated with cluster structure such that clustering results are easily comprehensible to users. Since a topic ontology provides an interpretation of a news topic at multiple levels of detail, an important use of a topic ontology is automatic cluster labeling. In addition, a topic ontology can be effectively used for suggesting alternative queries in information retrieval.

There exists research work on extraction of hierarchical relations between terms from a set of documents [17, 45] or term associations [46]. However, our work is unique in that the topical relations are dynamically generated based on incremental hierarchical clustering rather than based on human defined topics such as Yahoo directory (http://www.yahoo.com).

5 Experimental Results for Information Analysis

In this section, we present experimental results that demonstrate the effectiveness of the information analysis component. Section 5.1 illustrates our experimental setup. Experimental results are presented in Section 5.2.

5.1 Experimental Setup

For the empirical evaluation of the proposed clustering algorithm, approximately 3,000 news articles downloaded from CNN (http://www.cnn.com) are used. The

[8] There are two thresholds (for selectivity) that for ST (λ_1) and GT (λ_2), which are determined by experiments.

Sample topic	Sample events
Earthquake	Algeria earthquake, Alaska earthquake, Iran earthquake, etc
Flood	Russia flood, Texas flood, China flood, etc
Wildfire	Colorado wildfire, Arizona wildfire, New Jersey wildfire, etc
Airplane crash	Ukraina airplane crash, Taiwan airplane crash, etc
Court trial	David Westerfield, Andrea Yates, Robert Blake, etc
Kidnapping	Smantha Runnion, Elizabath Smart, Patrick Dennehy, etc
National Security	Mailbox pipebomb, Shoebomb, Dirty bomb, etc
Health	2002-West nile virus, 2003-West nile virus, SARS, etc

Table 7. Examples for selected topics and events

total number of topics and events used in this research is 15 and 180, respectively. Thus, the maximum possible number of clusters we can obtain (at level 1) is 180. Note that the number of documents for events ranges from 1 to 151. Table 7 illustrates sample examples for topics and events.

The quality of a generated cluster hierarchy was determined by two metrics, precision and recall. Let T_r be a class on topic/event r.[9] Then, a cluster C_r is referred to as a topic r cluster if and only if the majority of subclusters for C_r belong to T_r. The precision and recall of the clustering at level i (where K_i is the number of clusters at level i) then can be defined as follows:

$$P_i = \frac{1}{K_i} \sum_{r=1}^{K_i} \frac{|C_r \cap T_r|}{|C_r|} \tag{11}$$

$$R_i = \frac{1}{K_i} \sum_{r=1}^{K_i} \frac{|C_r \cap T_r|}{|T_r|} \tag{12}$$

Thus, if there is large topic overlap within a cluster, then the precision will drop down. Precision and recall are relevant metrics in that they can measure "meaningful theme". That is, if a cluster (C) is about "Turkey earthquake", then C should contain all documents about 'Turkey earthquake". In addition, documents, which do not talk about 'Turkey earthquake", should not belong to C.

[9] A class is determined by ground truth dataset. Thus, a class on topic/event r contains all documents on r, and does not contain any other document on other topics or events. In contrast, a cluster is determined by clustering algorithms. Note that there exists 1-1 mapping between event and cluster at level 1 of hierarchy, and topic and cluster at level 2 of hierarchy.

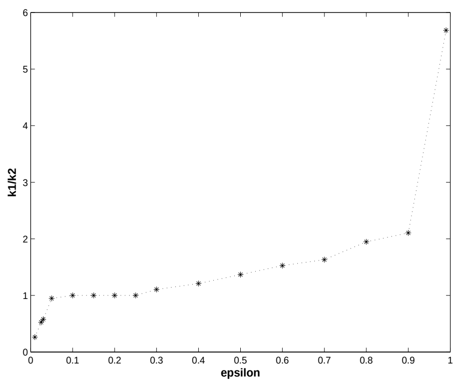


Fig. 5. Illustration of ϵ”’s sensitivity to clustering results

5.2 Experimental Results

For the purpose of comparison, we decided to use K-means clustering. However, since K-means is not suitable for incremental clustering, K-means clustering is performed retrospectively on datasets. In contrast, the proposed algorithm was tested on incremental datasets after learning IDF. Moreover, since we already knew the number of clusters at level 1 based on the ground-truth data, K could be fixed in advance. Furthermore, to overcome K-mean’s sensitivity to initial seed selections, a seed p is selected with the condition that the chosen seeds are far from each other. Since we deal with document datasets, the intelligent seed selection[10] can be easily achieved by using an inverted index.

Parameterization The size of a neighborhood, which is determined by ϵ, influences clustering results. To observe the effect, we performed an experiment as follows: From 3,000 documents, we organized sample datasets, which consists of 500 documents in 50 clusters of different sizes. Then, while changing the value of ϵ, our clustering was conducted on the dataset.

In Figure 5, the x-axis represents the value of ϵ, and the y-axis represents the number of clusters in the result ($k1$) over the number of clusters determined by ground-truth data ($k2$). Thus, if the clustering algorithm guesses the exact number of clusters, then the value of y corresponds to one. As observed in Figure 5, we could find the best result when ϵ varies between 0.1 and 0.25, i.e., the algorithm guessed the exact number of clusters. If the value of ϵ was too small, then the algorithm found a few large-size clusters. In contrast, many small-size clusters were identified if the value ϵ is too large. Thus, the proposed algorithm might be considered as sensitive to the choice of ϵ. However, once the value of ϵ (i.e., $\epsilon = 0.2$) was fixed, the approximately right number of clusters were

[10] Two documents are mutually orthogonal if they share no terms. This holds true when the Cosine metric is used for the similarity measure.

always obtained whenever we performed clustering on different datasets. Therefore, the number of clusters does not need to be given to our algorithm as an input parameter, which is a key advantage over partition-based clustering.

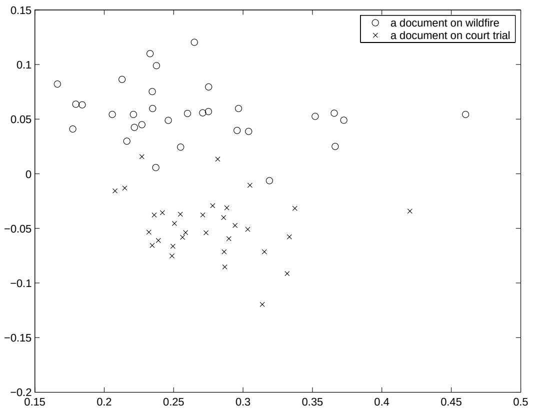


Fig. 6. Illustration of non-spherical document clusters

Ability to identify clusters with same density, but different shapes To illustrate the simple example for the shapes of document clusters with the same density, approximately the same number of documents were randomly chosen from two different events (a wildfire event and a court trial event), and the document×term matrix on this dataset is decomposed by Singular Value Decomposition. By keeping the first two largest singular values, the dataset could be projected onto a 2D space corresponding to principal components. Figure 6 illustrates the plot of the documents. As shown, since the shape of document cluster can be arbitrary, a shape of document cluster cannot be assumed in advance (e.g., hyper-sphere in k-means).

To test the ability of identifying the different shapes of clusters, we organized datasets where each cluster consists of approximately the same number of documents (but as illustrated in Figure 6, each document cluster will have a different shape). As shown in Figure 7, the proposed algorithm outperforms the modified K-means algorithm in terms of precision and recall.[11] This is because the proposed algorithm measures similarity between a cluster and a neighborhood of a document while K-means clustering measures similarity between a cluster and a document. Note that 10% increase in accuracy is significant by considering the fact that we provided the correct the number of clusters (K) and choose the best initial seed points for K-means.

[11] We did not compare the modified k-means algorithm with ours at level 2. To do this, we also need to develop a feature selection algorithm to extend the modified K-means algorithm into a hierarchical version.

	Precision	Recall
Level 1	91.5%	90.3%
Level 2	100%	76.4%

(a) Proposed algorithm

	Precision	Recall
Level 1	83.1%	86.7%

(b) Modified K-means

Fig. 7. Comparison of the clustering algorithms on datasets-1. Datasets-1 consists of five different datasets where each cluster has approximately the same density. The values of precision and recall shown in this table are obtained by averaging the accuracy of the algorithm on each dataset

Precision	Recall
87.5%	88.6%

(a) Proposed algorithm

Precision	Recall
78.7%	79.5%

(b) Modified k-means algorithm

Fig. 8. Comparison of the accuracy of clustering algorithms at level 1 on datasets-2. Datasets-2 consists of ten different datasets where each cluster has arbitrary numbers of documents. The values of precision and recall shown in this table are obtained by averaging the accuracy of the algorithm on each dataset

As illustrated in Figure 7, the recall of our algorithm decreases as the level increases. The main reason for this poor recall at level 2 is related to the characteristics of news articles. As discussed, a named entity (NE) plays a key role in defining who/when/where of an event. Hence, NE contributes to high quality clustering at level 1. However, at level 2, since the strength of topical terms are not very strong (unlike named entities), it was not easy to merge different event clusters (belonging to the same topic) into a same topical cluster.

Ability to discover clusters with different densities and shapes Since the sizes of clusters can be of arbitrary numbers, clustering algorithms must be able to identify the clusters with wide variance in size. To test the ability of identifying clusters with different densities, we organized datasets where each dataset consists of document clusters with diverse densities. As shown in Figure 8, when the density of each cluster is not uniform, the accuracy of the modified K-means clustering algorithm degraded. In contrast, the accuracy of our algorithm remains similar. Therefore, based on the experimental results on datasets-1 and datasets-2, we can conclude that our algorithm has better ability to find arbitrary shapes of clusters with variable sizes than K-means clustering.

Event confusion There are some events that we could not correctly separate. For example, on the wildfire topic, there exist different events, such as "Oregon wildfire", "Arizona wildfire", etc. However, at level 1, it was hard to separate those events into different clusters. Table 8 illustrates the reason for this event confusion at level 1. As shown, term frequency of topical terms (e.g., fire, firefighter, etc) is relatively higher than that of named entities (e.g., Colorado, Arizona, etc). Similarly, for the airplane crash topic, it was difficult to separate different airplane crash events since distinguishing lexical features like plane number has extremely low term frequency.

The capability of distinguishing different events on the same topic is important. One possible solution is to use temporal information. Rational behind this approach is based on the assumption that news articles on same event are temporally proximate, However, if two events occur during the same time interval, then this temporal information might not be helpful. Another approach is to use classification, i.e., training dataset is composed of multiple topic classes, and each class is composed of multiple events. After then, we learn the weight of topic-specific terms and named entities [49]. However, this approach is not relevant since we cannot accommodate the dynamically changing topics. Therefore, we need further study for the event confusion.

6 Conclusion and Future Work

We presented the mining framework that is vital to intelligent information retrieval. An experimental prototype has been developed, implemented and tested to demonstrate the effectiveness of the proposed framework. In order to accommodate topics that change over time, we developed the incremental document clustering algorithm based on a neighborhood search. The presented clustering algorithm could identify news event clusters as well as topic clusters incrementally. We also showed that presented topic ontologies could characterize news topics at multiple levels of abstraction.

We intend to extend this work into the following five directions. First, although a document hierarchy can be obtained using unsupervised clustering, as shown in Aggarwal *et al.* [1], the cluster quality can be enhanced if a preexisting knowledge base is exploited. That is, based on this priori knowledge, we can have some control while building a document hierarchy. Second, besides exploiting text data, we can utilize other information since Web news articles are composed of text, hyperlinks, and multimedia data. For example, as described in [25], both terms and hyperlinks (which point to related news articles or Web pages) can be used for feature selection. Third, coupling with WordNet [36], we plan to extend the topic ontology learning framework to accommodating rich semantic information extraction. To this end, we will annotate a topic ontology within Protégé [38, 54]. Forth, our clustering algorithm can be tested on other datasets like TDT corpus [53]. Finally, to strengthen our work in terms of generality, we are in the process of investigating the potential applicability of our method to earth science information streams.

	Colorado wildfire	Num	Arizona wildfire	Num
1	fire	14.33	fire	17.68
2	forest	5.72	*rodeo*	4.76
3	firefighter	4.83	blaze	4.38
4	acre	3.94	firefighter	4.21
5	evacuate	3.77	burn	3.92
6	*hayman*	3.22	*arizona*	3.46
7	blaze	3.11	*paxon*	3.15
8	weather	3.06	acre	3.00
9	official	2.89	wildfire	3.00
10	national	2.83	*chediski*	2.89
11	burn	2.72	resident	2.61
12	area	2.66	center	2.46
13	wildfire	2.56	national	2.46
14	*denver*	2.43	area	2.46
15	*colorado*	2.33	evacuate	2.65

Table 8. Top 15 high term frequency words in Colorado wildfire and Arizona wildfire event. Num represents the average number of term occurrences per document in each event (without document length normalization). Terms with italic font carry event-specific information for each wildfire event

7 Acknowledgement

This paper is based on our previous work [12], which was presented at the Second International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE 2003), Catania, Sicily, Italy, November 2003. We would like to thank the audience and anonymous reviewers of ODBASE 2003 for their helpful comments. We also would like to appreciate anonymous reviewers of this special issue for their valuable comments. Finally, we would like to thank Jongeun Jun for helpful discussions on the clustering algorithm.

This research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

References

1. C.C. Aggarwal, S.C. Gates, and P.S. Yu. On the merits of using supervised clustering for building categorization systems. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.
2. E. Agirre, O. Ansa, E. Hovy, and D. Martinez. Enriching very large ontologies using the WWW. In *Proceedings of the ECAI Workshop on Ontology Learning*, 2000.

3. R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence database. In *Proceedings of International Conference of Foundations of Data Organization and Algorithms*, 1993.
4. J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking: pilot study final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.
5. J. Allan, V. Lavrenko, and H. Jin. First story detection in TDT is hard. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management*, 2000.
6. N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322-331, 1990.
7. S. Berchtold, D.A. Keim, and H.P. Kreigel. The X-tree: An index structure for high dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, 1996.
8. M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573-595, 1995.
9. P.S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998.
10. T. Brants, F. Chen, and A. Farahat. A system for new event detection. In *Proceedings of the 26th International ACM SIGIR International Conference on Research and Development in Information Retrieval*, 2003.
11. K. Chan, and A.W. Fu. Efficient time series matching by wavelets. In *Proceedings of IEEE International Conference on Data Engineering*, 1999.
12. S. Chung, and D. McLeod. Dynamic topic mining from news stream data. In *Proceedings of the 2nd International Conference on Ontologies, Databases, and Application of Semantics for Large Scale Information Systems*, 2003.
13. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification (2nd Ed.)*. Wiley, New York, 2001.
14. D.M. Dunlavy, J. Conroy, and D.P. O'Leary. QCS: a tool for querying, clustering, and summarizing documents. In *Proceedings of Human Language Technology Conference*, 2003.
15. L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.
16. U.M. Fayyad, C. Reina, and P.S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1998.
17. E.J. Glover, D.M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, 2002.
18. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998.
19. S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, 1999.
20. A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*,

1985.

21. J. Han, and M, Kamber. *Data mining: concepts and techniques.* Morgan Kaufmann Publishers, 2000.
22. V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.
23. P. J. Huber. *Robust Statistics.* Wiley, New York, 1981.
24. R.A. Jarvis, and E.A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, C22, 1025-1034, 1973.
25. T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
26. G. Karypis, E.H. Han, and V. Kumar. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68-75, 1999.
27. L. Khan, and D. McLeod. Effective retrieval of audio information from annotated text using ontologies. In *Proceedings of ACM SIGKDD Workshop on Multimedia Data Mining*, 2000.
28. L. Khan, and D. McLeod. Disambiguation of annotated text of audio using onologies. In *Proceeding of ACM SIGKDD Workshop on Text Mining*, 2000.
29. L. Khan, D. McLeod, and E.H. Hovy. Retrieval effectiveness of an ontology-based model for information selection. *The VLDB Journal*, 13(1):71-85, 2004.
30. B. Larsen, and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.
31. X. Liu, Y. Gong, W. Xu, and S. Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th ACM SIGIR International Conference on Research and Development in Information Retrieval*, 2002
32. A. Maedche, and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72-79, 2001.
33. A. McCallum, K. Nigam, and L.H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.
34. K.R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J.L. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *Proceedings of the Human Language Technology Conference*, 2002.
35. I.D. Melamed. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Proceedings of the 3rd Workshop on Very Large Corpora*, 1995.
36. G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312, 1990.
37. R.T. Ng, and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, 1994.
38. N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, and M.A. Musen. Creating Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 6(12):60-71, 2001.

39. D. Pelleg, and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.
40. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
41. D.R. Radev, S. Goldensohn, Z. Zhang, and R.S. Raghavan. Newsinessence: a system for domain-independent, real-time news clustering and multi-document summarization. In *Proceedings of Human Language Technology Conference*, 2001.
42. D.R. Radev, S. Goldensohn, Z. Zhang, and R.S. Raghavan. Interactive, domain-independent identification and summarization of topically related news. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, 2001.
43. L. Ralaivola, and F. d'Alché-Buc. Incremental support vector machine learning: a local approach. In *Proceedings of the Annual Conference of the European Neural Network Society*, 2001.
44. G. Salton, and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.
45. M. Sanderson, and W.B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.
46. D. Song, and P. D. Bruza. Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology*, 54(4):321-334, 2003.
47. E.M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.
48. Y. Yang, J. Carbonell, R. Brown, T. Pierce, B.T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32-43, 1999.
49. Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
50. L.A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2):177-200, 1971.
51. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.
52. Y. Zhao, and G. Karypis. Evaluations of hierarchical clustering algorithms for document datasets. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management*, 2002.
53. Nist topic detection and tracking corpus.
http://www.nist.gov/speech/tests/tdt/tdt98/index.htm, 1998.
54. Protégé WordNet tab.
http://protege.stanford.edu/plugins/wordnettab/