Dynamic Topic Mining from News Stream Data

Seokkyung Chung and Dennis McLeod

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

Abstract. Given the popularity of Web news services, we propose a topic mining framework that supports the identification of meaningful topics (themes) from news stream data. 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 documents update. That is, since several hundred news articles are published everyday by a single Web news service, it is essential to develop incremental data mining tools to cope with such dynamic environments. To this end, we present a sophisticated incremental hierarchical document clustering algorithm using a neighborhood search. The novelty of our proposed algorithm lies in exploiting locality information to reduce the amount of computation while producing high-quality clusters. Other components of topic mining (e.g., learning topic ontologies) can be performed based on the obtained document hierarchy. Experimental results show that our proposed incremental clustering produces high-quality clusters, and topic ontology provides an interpretation of the data at different levels of abstraction.

1 Introduction

As every news organization provides Web news service, Internet users are experiencing overwhelming quantities of online news information. In these circumstances, the users have no time to read every news story. Instead, the navigation of news repository should be aided by data mining tools that allow the users to quickly locate their information needs from a news stream.

Web news articles are composed of hyperlinks, audio, video, images, and text. However, since not all news stories have corresponding multimedia data, text carries the most important information about the news. Since text is unstructured data, efficient text mining and access methods are required to obtain valuable knowledge embedded into the text document.

The simplest document access method is to employ keyword-based retrieval. Although this method seems to be effective, it suffers from serious drawbacks. For example, if a user chooses irrelevant keywords (due to his/her vague information

needs or unfamiliarity with the domain of interest), retrieval accuracy will be degraded. In addition, since keyword-based retrieval is based on simple keyword counting, it cannot address language semantics (e.g., synonym).

The problems stated above have been tackled by query expansion based on domain-independent (general) ontologies like WordNet [14]. However, it is well known that this approach leads to a degradation of precision. That is, since the words introduced by term expansion may have more than one meaning, using additional terms can improve recall, but decrease precision. Manually building ontology with controlled vocabulary is helpful in this situation [10]. However, though ontology authoring tools have been developed in the past decades, constructing ontology by hand whenever we encounter new domains is error-prone and time-consuming work. When developing information retrieval applications based on ontologies, this knowledge acquisition bottleneck problem is always faced. Therefore, integration of knowledge acquisition with data mining, which is referred to as ontology learning, becomes a must [12].

To illustrate our sample application within news stream data management, consider the following two issues. First, most of the news articles are related to previously published stories or future ones. Hence, this strong temporal dependency between the documents should not be ignored when navigating the news collection. In some Web news services (e.g., Washington Post), in order to present comfortable navigation interfaces to the user, editors manually add hyperlinks to the related stories when publishing news articles. Thus, a story about the safe return of a kidnapped child can be linked to the earlier stories reporting the police's search for the abducted child, investigation of the suspect, etc. Since those hyperlinks are added at the time of publication, they always point backwards in time. However, to read subsequent stories, a news service system should provide hyperlinks which allow a user to navigate forward in time. Next, search engines in current Web news services can only find the users' expected information. This is because the specified keywords are generated from their knowledge space. Note that the information the users cannot specify may be valuable to them. For example, if the users do not know about an airplane crash that occurred yesterday, then they cannot issue a query about that accident though they might be interested in that topic.

The first issue can be addressed by considering the nearest neighbors. A simple distance metric between two document feature vectors will show how much two documents are related to each other. Hence, this nearest neighbor analysis allows us to automatically identify related stories. Moreover, it can be used to find near-duplicate articles. This especially holds true for news feeds that tend to repeat stories with minor changes from hour to hour. In this situation, presenting the only most recent article is probably sufficient.

In addition, if the documents are clustered according to their topics and the meaningful labels are assigned to each cluster, then those labels can be used to understand the main themes of the clusters. Hence, if a user found an interesting story about a famous actor's court trial, and wants to know about the movies he stars in, then the user can jump from the "court trial" cluster

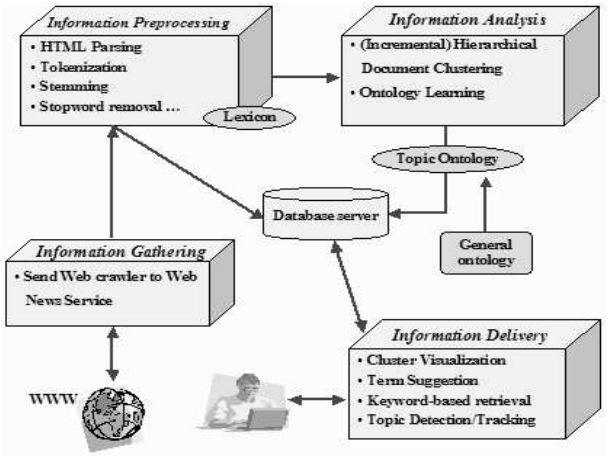

Fig. 1. Overview of a proposed model

to the "entertainment" cluster based on the cluster labels. Moreover, if a user has vague information needs, document cluster hierarchy can be a reasonable starting point.

To address the above issues, we propose a topic mining framework that supports the identification of meaningful topics (themes) from news stream data. News articles are retrieved[1] 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[2]. Fig. 1 illustrates main parts of topic mining.

The challenging issue within news repository management is the high rate of documents insertion. That is, since several hundred news stories are published everyday by a single Web news service, it is essential to develop incremental data mining tools to cope with such dynamic environments. To this end, we present a sophisticated incremental hierarchical document clustering algorithm using a neighborhood search. The novelty of our proposed algorithm lies in exploiting locality information to reduce the amount of computation while producing high-quality clusters. Other components of topic mining can be performed based on the obtained document hierarchy.

The remainder of this paper is organized as follows. Section 2 presents the

[1] In the information gathering stage, a Web crawler retrieves a set of news stories 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 news Web site on a daily basis.

[2] Due to the space limitation, we skip the detailed discussion about information delivery.

related work on document clustering. Section 3 provides the discussion about the information preprocessing step, and Section 4 explains information analysis, which is a key focus of this paper. Finally, we conclude the paper and provide our future plans in Section 5.

2 Related Work

Utilizing document clustering for intelligent information retrieval is not a new idea. In this section we provide a brief survey on previous text clustering work.

Document clustering has received significant attention during the past decades [6, 7, 9, 11, 1, 22, 15]. This work is broadly classified into two categories: center-based clustering and hierarchical clustering.

Center-based algorithms find the clusters by partitioning the entire dataset into either a pre-determined (e.g., k-means clustering) or an automatically derived number of clusters (e.g., X-means clustering) [6, 11, 15]. A key characteristic of many center-based clustering algorithms is that they use a global criterion function whose optimization produces the entire clustering process. The main disadvantage of this approach is that it is susceptible to a local optimum.

In contrast, 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 [6, 22]. Thus, its result is in the form of tree, which is referred to as a *dendrogram*. The main advantage of HAC lies in its ability to provide a view of data at multiple levels of abstraction. However, we should determine where to cut the dendrogram to produce clusters. This step is usually performed by human visual inspection, which is a time-consuming and subjective process.

To cope with frequent document insertion into the database, incremental clustering algorithm should be applied to news repository management [21, 2]. However, previously proposed algorithms on incremental clustering are not applicable to our topic mining framework in that (1) they are sensitive to the input order (i.e., it produces different clusters for different order of the same input data); or (2) they are not relevant for high-dimensional data like documents.

3 Information Preprocessing

The information preprocessing step extracts meaningful information from the collected data using NLP tool, and transform unstructured text data into structured knowledge. Toward this end we employ natural language processing tools as follows:

- *HTML preprocessing.* Since downloaded news articles are in HTML format, we remove irrelevant HTML tags, and extract meaningful information.
- *Tokenization.* Its main task is to identify the boundaries of the words.

- *Stemming.* There can be different forms for the same words (e.g., *students* and *student*, *go* and *went*). We need to convert these different forms of the same word to their root. Toward this end, instead of solely relying on Porter stemmer [16], in order to deal with irregular plural/tense, we combine Porter stemmer with the lexical database [13].
- *Stopwords removal.* Stopwords are the words that occur frequently in text but do not carry useful information. For example, *have*, *did*, and *get* are not meaningful. Removing such stopwords provides us with a dimensionality reduction effect. We employ stopword list used in Smart project [18].

After preprocessing, a document is represented as a vector in an n-dimensional vector space [18]. The simple way to do this is to employ *bag-of-words* approach. That is, all content-bearing words in the document are taken and any structure of text or the word sequence is ignored. We treat each term as a feature, and represent each document as a vector of certain weighted word 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 following two heuristics.

- Important terms occur more frequently within a document than unimportant terms do.
- The more times a term occur 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 length. Let $freq_{ij}$ be the number of w_i's occurrence in a document j, and l be the length of the document j. Then, term frequency (tf_{ij}) of w_i in the document j is defined as follows:

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

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

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

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

We refer to the above ranking scheme as a static TF-IDF since it is based on static document collection. However, since our data is incrementally updated, we learn *idf* from certain amount of documents (i.e., the document frequency is generated from training corpus), and incrementally update *idf* as we process subsequent documents. In particular, we employ an incremental update of the *idf* value proposed by Yang et al. [19].

Finally, to measure closeness between two documents, we employ Cosine metric which has been widely used in much information retrieval literature [18]. It

	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. Document × term matrix

measures similarity of two items according to the angle between them. Thus, vectors pointing to similar directions are considered as representing similar concepts. The cosine of the angles between two n-dimensional vectors x and y is defined by

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

4 Information Analysis

In the information analysis step, we perform incremental hierarchical document clustering to build dynamic document cluster hierarchy. In addition, an algorithm for dynamic topic ontology learning is provided. In this step, we can also incorporate background knowledge (e.g., WordNet) in order to reinforce our automatically obtained topic ontology.

Section 4.1 explains the motivation of our proposed clustering algorithm. Section 4.2 provides a non-hierarchical incremental clustering algorithm using a neighborhood search. In Section 4.3, we show how to extend it into hierarchical version. Finally, in Section 4.4, we discuss how to build a dynamic topic ontology.

4.1 A Motivating Example

In this section, we provide the basic motivations for the proposed clustering algorithm. In particular we explain why neighborhood-based approach is useful in incremental document clustering.

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

- D_1: A child is kidnapped so police starts searching.
- D_2: Police found the suspect of child kidnapping.
- D_3: An abducted boy safely returned home.

In the above three documents, though D_1 and D_2 is similar, and D_2 and D_3 is similar, D_1 and D_3 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.

- *Fuzzy similarity relation.* As discussed in the fuzzy theory [20], the similarity relation does not satisfy transitivity. To make it satisfy transitivity, fuzzy transitivity closure can be used. However, this is not scalable with the number of data points.
- *Inherent characteristic of news.* As discussed in Allan et al. [2], *event* is defined as "some unique thing that happens at some point in time". Thus, event is considered as an evolving object through some time interval. Hence, though the documents belong to the same event, if they discuss different aspects of the event, the words they use should be different.
- *Language semantics.* We need to consider the diversity of word usage for the same meaning. Using only keyword counting aggravates the problem.

The transitivity is important in the incremental clustering algorithm. Consider the following incremental clustering algorithm which has been widely used in Topic Detection and Tracking research [2].

1. Initially, there is only one news story, and it forms a singleton cluster.
2. For an incoming story (d_i), we compute the similarity between d_i and pre-generated clusters. The similarity is computed by the distance between d_i and the representative (e.g., center) of the cluster.
3. Selects the cluster (C_i) which has the maximum proximity with d_i.
4. If the similarity between d_i and C_i exceeds a pre-defined threshold, then all documents in C_i are considered as related stories to d_i (topic tracking), and d_i is assigned to C_i. Otherwise, d_i is considered as a novel story (first story detection), and creates a new cluster for d_i.
5. Repeat 2-4 whenever a story comes out.

The above algorithm implies that the order of document insertion is critical. For example, in Table 1, if the order of document insertion is "$D_1 D_2 D_3$", then we may obtain one cluster $\{\{D_1, D_2, D_3\}\}$. However, if the order is "$D_1 D_3 D_2$", then two clusters will be obtained: $\{\{D_1, D_2\}, \{D_3\}\}$. Though the order of document insertion is fixed (because the document is inserted into a database whenever they are published), it is undesirable if the clustering result heavily depends on the insertion order.

In what follows, we identify two key requirements that the incremental document clustering should satisfy:

1. *Insensitivity to the input order.* Given different ordering of same dataset, many incremental clustering algorithms produce different clusters, which is an unreliable phenomenon. Hence, the incremental clustering should be robust to the input sequence. Thus, for the example above, regardless of the input order, the successful algorithm should produce a single cluster, $\{\{D_1, D_2, D_3\}\}$.
2. *Fast incremental processing of a new document.* Due to the frequent document insertion into the database, triggering the whole clustering process is computationally infeasible. Hence, to cope with such dynamic nature of the problem, the placement of a new document should not be decided by

Notation	Meaning
n	The total number of documents in a database
d_i	A new document
ϵ	Threshold for determining the neighborhood
$N_\epsilon(p)$	ϵ-neighborhood for p
D_d	The set of documents which contain any term of d
C_d	The set of clusters which contain any neighbor of d
$\|A\|$	The size of a set A
S_j	Signature vector for a set A_j
df_i^j	Document frequency of w_i within a set A_j
s_i^j	$j-$th component of S_i

Table 2. Notations for incremental non-hierarchical document clustering

a metric that requires all the documents that have been processed in the past. Whenever a new document is inserted, it should perform fast update of existing cluster structure.

4.2 Incremental Non-hierarchical Document Clustering

In this section, we propose incremental non-hierarchical document clustering algorithm based on a neighborhood search. Before going into detailed discussions, we present definitions for necessary terminologies. Table 2 also outlines the symbols with the meaning, which will be used throughout this paper.

Definition1 similar. If $similarity(p, q) \geq \epsilon$, then a point p is referred to as similar to a point q.

Definition2 $N_\epsilon(p)$**.** ϵ-neighborhood for a point p is $\{x : similarity(x, p) \geq \epsilon\}$.

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

Our proposed clustering algorithm exploits characteristic of a neighborhood. The idea is that label of an object is influenced by the features of its neighbors. Examples of such features are the labels of its neighbors, or the percentage of the neighbors which satisfy the certain constraint. Based on this idea, we define *Cluster Membership Hypothesis* as follows:

Definition3 Cluster Membership Hypothesis. A point p is referred to as belonging to a cluster C_i if and only if 1 - δ fraction of $N_\epsilon(p)$ belong to C_i.

Fig. 2 illustrates the proposed incremental clustering algorithm. The overall algorithm proceeds in four phases: initialization, neighborhood search, identifying an appropriate cluster for a new document, and re-clustering based on local information.

Step 1. Initialization:
Document d_0 forms a singleton cluster c_0;

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

Step 3. Identifying a cluster which can host a new document:
Let C_{d_i} be the the set of clusters containing
any document belonging to $N_\epsilon(d_i)$;
Compute similarity between d_i and a cluster $c \in C_{d_i}$

Based on the value obtained from above,
if there exists a cluster which can absorb d_i, then
    add d_i to the cluster and update DCF vector for the cluster;
otherwise,
    create a new cluster for d_i and
    create a corresponding DCF vector for this new cluster;

Step 4. Re-clustering:
Let C_i be the cluster which hosts d_i;
If C_i is not a singleton cluster, then trigger merge operation;

Fig. 2. The incremental non-hierarchical document clustering algorithm

Step 1: Initialization. Initially, we assume that only one document is available. Hence this document itself forms a singleton cluster.

Step 2: Neighborhood search. Achieving an efficient neighborhood search is a key issue in our proposed clustering algorithm. If the dimensions of the data are low (e.g., 5-20), then the multi-dimensional index structure can be employed to support fast search [3, 8, 4]. However, since our data set has extremely high dimensions, though we conduct dimensionality reduction using Singular Value Decomposition [5], the reduced dimensions (which are more than 100) are still beyond the capacity of the current spatial indexing structure.

Instead we employ an inverted index for the purpose of the neighborhood search. In the inverted index [18], the index associates a set of documents with words. That is, for each word w, we build a document list which contains all documents containing w. Let a document d be composed of $w_1, \ldots, w_n$. In order to

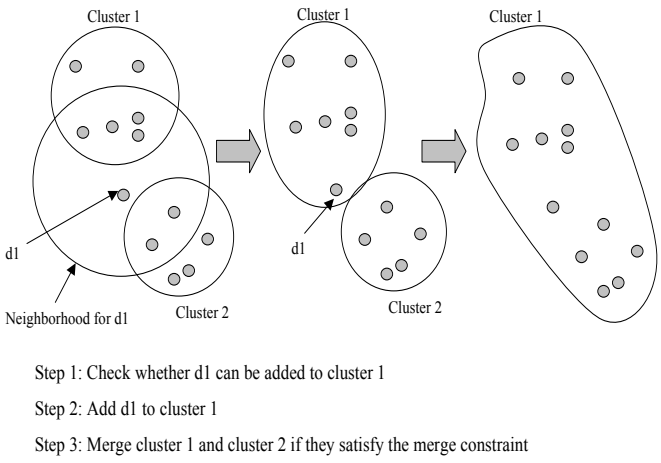

Fig. 3. Illustration of re-clustering phase

find similar documents to d, instead of looking into whole document collections, it is sufficient to examine the documents which contain any w_i. Hence, given document d, finding the neighborhood can be accomplished in $O(|D_d|)$ where D_d is the set of documents containing any w which appear in d[3].

Step 3: Identifying an appropriate cluster. To assign an incoming document (d_i) to the existing cluster, we need to compute the similarity between each candidate cluster and a document. Using the neighborhood of the new document, we determine the cluster which can host a new document.

One possible approach is to select the cluster that has the biggest number of its members in the ϵ-neighborhood. This approach only considers the number of documents in the overlapped region, and ignores the proximity between neighbors and the d_i.

Instead, we can exploit the similarity between the candidate cluster and its neighbors when determining the assignment. That is, 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. Then, we select a cluster for the new document which has the maximum value for the following formula:

$$\sum_{n_j \in N_\epsilon(d_i)} w_j \cdot Similarity(n_j, S_k) \text{ where } S_k \in C_d \tag{4}$$

In addition, we can simply measure the similarity between the signature vector of the neighborhood and that of the cluster.

[3] Note that $|D_d|$ is much smaller than n thanks to Zipf's law [17].

Then, how can we define a signature vector? The signature vector should be composed of words which reflect the main characteristics of the documents in the set. For each word w_i in the set A_j (e.g., a cluster/neighborhood), we compute the weight for the word using a following formula:

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

In Equation (5) the first factor measures document frequency within a set, and the second factor measures the sum of the weight for the word over the whole documents within a set.

When computing similarity between a document and a cluster, we only need signature vectors of a cluster and a document. This is a computationally efficient scheme if the signature vector can be incrementally updated as a new document is added to the cluster. The notion of Document Cluster Feature (DCF) vector is defined for this purpose.

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 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.

Based on the definition of DCF and above theorem, we can incrementally update the signature vector of a cluster whenever a new document is inserted. Therefore, if there exists a cluster (say, C_i) which can absorb d_i (i.e., the similarity is greater than pre-defined threshold), then assign it to C_i, and update the DCF_i. Otherwise, create a new cluster for d_i and DCF vector for this cluster.

Step 4: Re-clustering. If we assign d_i to C_i, then we need to trigger the merge operation. This is based on local approach. Instead of re-clustering the whole data set, we only need to focus on the clusters which are affected by the new document. That is, a new document is placed in the cluster, and a sequence of cluster re-structuring process is performed only in regions that have been affected by the new document. Fig. 3 illustrates this idea. As shown, we only consider clusters which contain any document belonging to the neighborhood of a new document.

Recent data mining literature have proposed Shared Nearest Neighbors (SNN) clustering approach [7, 9]. In this method, the k-nearest neighbors of each point are identified, and similarity between two points is defined as the number of

neighbors they share. Thus, basic motivation of SNN clustering is similar to ours, however, the detailed approach is completely different. The key difference between SNN and ours is how to define a neighborhood. In SNN, a neighborhood is defined as a set of k-nearest neighbors while we employ an ϵ-neighborhood concept. Thus, the neighborhood constructed from k-nearest neighbors is local in that it is defined narrowly in dense region, and more widely in sparse region. However, for document clustering, a global neighborhood approach produces more meaningful clusters. Next, SNN is defined in the static dataset while ours is working on incremental dataset. Finally, our algorithm can easily identify the singleton clusters. This is especially important in our application since unique documents in a news stream may imply new events that have not been mentioned in previous articles. In contrast, SNN considers singleton clusters as noise, and remove them, hence it does not cluster all the points.

4.3 How to extend Non-hierarchical Clustering into Hierarchical Version?

When we apply the algorithm in Fig. 2 to the news story collection, we can obtain different types of event clusters. These clusters are defined at level 1. Note that level 0 corresponds to the lowest level in a cluster tree (i.e., each document itself is a cluster at level 0). Now, since our goal is to generate document hierarchy, all trial clusters at level 1 should be combined as a single cluster at level 2 to reflect the court trial topic. However, due to the named entity (NE)[4], this becomes a difficult task since NE has very high term-frequency within a document. Consequently, similarity between different event clusters which belong to same topic becomes extremely low. In this section, we provide a hierarchical clustering algorithm, and show some preliminary experimental results. Before presenting our algorithm, we first define necessary terminologies in the following.

Definition 6. [Specific features (SF)] Specific features for a cluster C_i is the collection of terms which frequently occur within a cluster C_i, but rarely occur outside of C_i. We denote it by SF_{C_i}.

Definition 7. [Actual time (T)] Actual time is obtained from the time stamp of each news article. For example, T corresponds to the date of the publication for a news article.

Definition 8. [Virtual time (t)] Virtual time t is initialized by 0. At any time t, only one operation (e.g., document add, cluster merge, etc) can be performed. In addition, time is increased by one when we perform an operation and remains unchanged if there is no operation.

Let $df^i(t)$ be the document frequency of a word w_i in whole documents at time t. Then, document frequency of w_i at time $t+1$ is defined as follows:

$$df^i(t+1) = \begin{cases} df^i(t)+1, & \text{if } d \text{ is added at } t \text{ and it contains } w_i \\ df^i(t), & \text{otherwise} \end{cases} \quad (6)$$

[4] Named entities are people/organization, time/date and location, which play a key role in defining "who", "when", and "where" of an event.

Let $df_{in}^{ij}(t)$ be the document frequency of a word w_i within a C_j at time t. Then $df_{in}^{ij}(t+1)$ is recursively defined as follows:

$$df_{in}^{ij}(t+1) = \begin{cases} df_{in}^{ij}(t)+1, & \text{if } d \text{ is added to } C_j \text{ at } t \text{ and } d \text{ contains } w_i \\ df_{in}^{ij}(t), & \text{otherwise} \end{cases} \quad (7)$$

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

$$k(t+1) = \begin{cases} k(t), & \text{if } d \text{ is added to an existing cluster at } t \\ k(t)+1, & \text{if } d \text{ itself forms a new cluster at } t \\ k(t)-1, & \text{if two clusters are merged at } t \end{cases} \quad (8)$$

Let $df_{out}^{ij}(t+1)$ represent how much w_i occurs outside the cluster C_j at time $t+1$.

$$df_{out}^{ij}(t+1) = \frac{df^i(t+1) - df_{in}^{ij}(t+1)}{k(t+1)} \quad (9)$$

Then, predictive strength of a word w_i for the cluster C_j at time $t+1$ is defined as follows:

$$pred_j^i(t+1) = log \frac{df_{in}^{ij}(t+1)}{df_{out}^{ij}(t+1)} \quad (10)$$

In sum, Equation (10) gives more weight to the words occurring frequently within C_j and less weight to those occurring rarely outside of C_j. Thus, a highly predictive word can select the most specific feature for the cluster.

We informally describe our hierarchical algorithm as follows. We generate clusters at level 1 based on the algorithm in Fig. 2. During some pre-defined time interval, if no more documents are added to certain cluster at level 1, then we assume that the event for the cluster ends[5], and associate SF with this cluster at level 1. We then perform neighborhood search for this cluster to generate cluster at level 2. Since SF reflects the most specific characteristics for the cluster, it is not helpful to merge two topically similar clusters. Hence, when we build a vector for C_i^j, terms in SF (for C_i^j) are not included for building cluster vector.

Our clustering algorithm exploits multi-resolution inverted index for the efficient discovery of neighborhood at different levels of the hierarchy. That is, in the inverted index at level i, the index associates a set of clusters (at level i) with words. For each word w, we build a cluster list which is composed of all clusters containing w. Note that the cluster is referred to as containing a word w if the second factor of Equation (5) exceeds the pre-defined threshold.

For the empirical evaluation of the proposed clustering algorithm, we manually labeled approximately 2,000 stories downloaded from CNN news Web site. The total number of topics and events used in this research is 15 and 150, respectively. Thus, the maximum possible number of clusters we can obtain (at

[5] This assumption is based on the temporal proximity of the event [19].

	Precision	Recall
Level 1	91.5%	90.3%
Level 2	100%	81.4%

(a) Proposed algorithm

	Precision	Recall
Level 1	83.1%	86.7%

(b) Modified K-means

Fig. 4. Comparison of the clustering algorithms

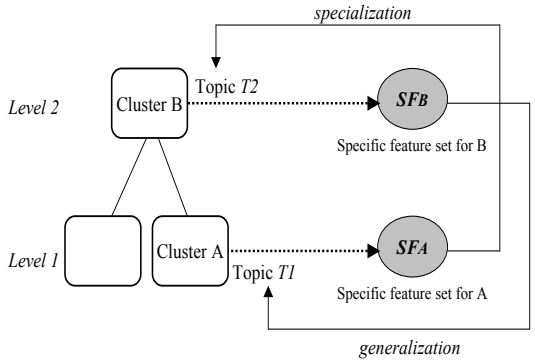

Fig. 5. Illustration of relationship in topic ontology

level 1) is 150. Note that the number of stories for the events ranges from 1 to 142. The quality of clustering solution was determined by two metrics, precision and recall. Let T_r be a class on topic r. Then, 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:

$$P_i = \frac{1}{k_i} \sum_{j=1}^{k_i} \frac{|C_j \cap T_j|}{|C_j|} \tag{11}$$

$$R_i = \frac{1}{k_i} \sum_{j=1}^{k_i} \frac{|C_j \cap T_j|}{|T_j|} \tag{12}$$

Fig. 4 compares the accuracy of our clustering algorithm with the modified k-means algorithm. Since k-means is not suitable for incremental clustering, we performed k-means clustering retrospectively on the dataset while our proposed algorithm was tested on incremental data set after learning idf. Since we already know the number of clusters at level 1 (based on the ground-truth data), for k-means, we fixed k in advance. In addition, to overcome the k-mean's sensitivity to the initial seed selections, we select the seed p with the condition

Event	Specific features
Court trial 1	winona ryder actress shoplift beverly
Court trial 2	andrea yates drown insanity
Court trial 3	blake bakley actor
Kidnapping 1	elizabeth smart utah salt lake
Kidnapping 2	jessica holly soham cambridgeshire england
Kidnapping 3	weaver ashlei miranda gaddis
Wildfire	wildfire firefighter colorado forest blaze acre

Table 3. Examples of specific features for the clusters at level 1

that the chosen seed should not belong to the neighborhoods of the pre-selected seeds. Each seed selected by the above process is far from each other since they are approximately mutually orthogonal. As shown in Fig. 4, the proposed algorithm outperforms the modified k-means in terms of precision and recall[6]. As illustrated, the recall of our algorithm decreases as the level increases. The main reason for this poor recall at level 2 is related with the characteristics of the news articles. As discussed, named entity plays a key role in defining who/when/where of an event. Thus, at level 1, NE contributes to high quality clustering. However, at level 2, since the strength of topical terms are not so strong like named entity, it is not easy to merge different event clusters (belong to the same topic) into same topical cluster. There are some events 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, we failed to separate those events into different clusters.

4.4 Building Topic Ontology

One of the main problems with domain-independent ontologies is that topically related concepts/terms are not explicitly linked. That is, there is no relationship between *court-attorney*, *kidnap-police*, etc.

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

A topic ontology is built on top of the document hierarchy. We associate each node of hierarchy with specific features. Specific features of a parent node are more topically abstract than those of a child node. That is, as shown in

[6] Note that we did not compare modified k-means with ours at level 2. To do this, we also need to develop feature selection algorithm to extend modified k-means into hierarchical version.

Topic	Specific features
Earthquake	quake earthquake magnitude collapse building
Court trial	court trial law murder defense legal prosecutor attorney testify jury evidence kill
Kidnapping	girl miss parent disappear abduct enforce police family kidnap
Airplane crash	crash air airline safety dead boeing accident traffic passenger flight pilot collision aircraft warn aviat
West nile virus	disease blood organ louisiana symptom spread
Wildfire	arizona burn evacuate hayman rodeo residence wind denver force size danger flame national line

Table 4. Examples of specific features for the clusters at level 2

Event	General features
Trial 1	hear hill gerago women delay injury guilty camera store stand charge court target trial drug arm count law legal attorney evidence victim arrest order

Table 5. Examples of general features for the court trial cluster 1

Table 3 and Table 4, with respect to the topic, 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. Hence, SF at lower level is a specialization of a higher level node while SF at upper level is a generalization of a lower level node. Fig. 5 illustrates this idea. SF_A is a specialization with respect to the topic T_2 while SF_B is a generalization of the topic T_1. Thus, SF_B is a super concept of SF_A with respect to topic T_1.

We can also generate general features (GF) for the node which is defined as follows:

Definition 9. [General features (GF)] General features for a cluster C_i is the collection of terms which frequently occur within a cluster C_i, and also occur outside of C_i. We denote it by GF_{C_i}.

In comparison with SF, the predictive power of GF is weaker than that of SF. Table 5 shows GF for a cluster for "court trial 1" in Table 3. In sum, those SF and GF constitute the concepts of topic ontology.

5 Conclusion and Future Works

This paper presented a topic mining framework that is vital to intelligent information retrieval. In order to accommodate dynamically changing topics, the incremental document clustering based on neighborhood search is employed. We also presented topic ontology learning scheme based on the obtained document hierarchy.

We intend to extend this work in the following directions. Though we can obtain document hierarchy based on unsupervised clustering, as shown in Aggarwal et al. [1], we can enhance the cluster quality if we exploit the pre-existing knowledge base. In addition, we can augment topic ontology using WordNet to obtain rich semantics.

6 Acknowledgement

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. 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.
3. 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.
4. 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.
5. 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.
6. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Ed.)*. Wiley, New York, 2001.
7. 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.
8. A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1985.
9. 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.
10. L. Khan. Ontology-based information selection. *Ph.D. Thesis, University of Southern California*, 2000.

11. 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.
12. A. Maedche, and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), 2001.
13. 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.
14. G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312, 1990.
15. 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*.
16. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
17. M. Sahami. Using machine learning to improve information access. *Ph.D. Thesis, Stanford University*, 1999.
18. G. Salton and M.J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.
19. 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: Special Issue on Applications of Intelligent Information Retrieval, 14(4):32-43*, 1999.
20. L.A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3:177-200, 1971.
21. 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.
22. 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.