Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

A Dynamic Graph Model for Analyzing Streaming News Documents

Elizabeth Leeds Hohman
Naval Surface Warfare Center
Dahlgren, VA 22448
Email: elizabeth.hohman@navy.mil

David J. Marchette
Naval Surface Warfare Center
Dahlgren, VA 22448
Email: david.marchette@navy.mil

***Abstract*— In this paper we consider the problem of analyzing streaming documents, in particular streaming news stories. The system is designed to extract statistics from the document, incorporate these into a graph-based model, and discard the document to reduce storage requirements. The model is defined in terms of a changing lexicon and sub-lexicons at each node in the graph, with the nodes of the graph representing topics. An approximation to the TFIDF term weighting is introduced. We illustrate the methodology on a dataset of news articles, and discuss the dynamic nature of the model.**

I. Streaming Documents

Much research has been performed on text processing and there are many effective methods for representing and classifying a corpus of text documents into a set of pre-defined categories. (A detailed overview of text categorization is provided by [13].) Some methods have also been developed for clustering documents when the categories or cluster topics are not pre-defined ([6], [4], [8], and [11]).

Most text processing tasks assume that all documents in the corpus are available at once. Less work has been performed on text processing under the assumption that documents will continue to be presented. There are many types of evolving text sources available electronically such as on-line news sources and web logs. The quantity and nature of this text makes this an important area of research. We will refer to this problem as streaming document processing. The Topic Detection and Tracking (TDT) program [1] refers to the task of clustering documents in this environment as *cluster detection*. As discussed in [2], determining that a new cluster should be created and evaluating the results of clustering in this context is a difficult and ongoing research area.

We should explain the difference between this and "streaming text". In the case of streaming text, the text is being presented in a streaming manner with no delineation between chunks of text. For example, someone typing at a keyboard or transcripts from conversations or news feeds. In this case, the processing occurs as each word is presented and before the full document is available. So each word or phrase can be thought of as an observation. In the case of streaming documents, each document is considered an observation, whether that be a news article, an email, a web log entry, or some other text observation. Any processing is performed on the full document.

The volumes of text data that can be considered in this way can vary greatly. News articles can be collected in several hundred to thousand per day while emails occur at several hundred per day (for an individual) to several thousand a day (at a server). Web postings may be in many thousands or even millions per day if one considers web logs, newsgroups, chats, and other sources. Other cases of streaming documents include incidence reports at a major computer vendor, air traffic reports, and hospital admittance reports (for detecting outbreaks). These are all examples of streaming documents that can range from several hundreds to thousands to millions per day depending on the level of data collection.

A vector space model is usually used to represent text documents [12]. This representation casts each document as a vector with length equal to the number of words in the lexicon and with each vector entry corresponding to a weight for one of the words in the document. Words are usually weighted by the frequency of the word in the corpus. This frequency must be approximated in the case of streaming documents since the corpus is not fixed.

In Section II, we will briefly explain the vector space model and the need for an approximation to the model for the streaming case. We will introduce a method for approximating the standard TFIDF-weighted vectors in the vector space model. In Section III, we will describe a graph model for representing a streaming collection of documents. We describe the experimental data in Section IV and attempt to visualize the data and the graph clustering in Section V. Sections VI and VII explain some weaknesses of the model and discuss future work to address these issues.

II. Text Representation

Most text clustering and categorization methods use a vector space representation of documents [12]. Each document is represented as a vector, $\mathbf{x}_d \in \mathbb{R}^L$, $d = 1, ..., D$ where D is the number of documents in the corpus and L is the number of words in the lexicon. Each dimension of the vector corresponds to a different word in the lexicon. Let x_{dj} be the j^{th} entry of $\mathbf{x}_d$. Its value depends on the number of times the j^{th} word in the lexicon occurs in document d as well as its rate of occurrence in the corpus.

If we no longer assume a fixed corpus, we cannot use this fixed-dimensional vector space model. Since the lexicon

is constantly changing, we cannot pre-assign dimensions of the vector space to specific words. Since the lexicon cannot grow without limit, approximations must be made to the representation. New words will appear in documents while words that have been seen in the past might not be seen again. The approach we take in this work is to assign a counter to each word in the lexicon and update the counter each time the word is seen in a document. Once the lexicon reaches a pre-defined upper limit, a percentage of words can be removed from the lexicon where the words chosen for removal are those with the smallest counter value. In this way, words that fall out of use or are the result of typographical errors will be removed. However, if a new document reintroduces a word to the lexicon, the comparison of the new document to old documents will assume (not always correctly) that the word was not in the old documents.

Since vector entries are dependent not only on the frequency of the word in the document but also on the frequency of the word in the corpus, the corpus frequency must be approximated as well in the case of streaming documents. These details are discussed in the next Section.

A. Frequency-based word weighting

In the vector representation of a document, the j^{th} entry of the vector depends on the number of times the j^{th} word in the lexicon occurred in the document. The value is usually the occurrence of the word multiplied by a word weight which is dependent on the frequency of the word in the corpus. This results in high-frequency but context-independent words such as "the", "and", etc. having low values.

The most common weighting method is Term Frequency Inverse Document Frequency (TFIDF) [12]. Let TF_{ij} be the frequency of word j in document i and let IDF_j be the inverse document frequency of word j in the corpus. That is,

$$TF_{ij} = \frac{n_{ij}}{\sum_{k=1}^{D} n_{kj}}$$

where n_{ij} is the number of times word j occurred in document i and

$$IDF_j = \frac{D}{b_j}$$

where D is the number of documents in the corpus and b_j is the number of documents that contain word j. Then the TFIDF weight for word j in document i is given by

$$x_{ij} = TF_{ij} \log(IDF_j). \tag{1}$$

The x_{ij} become the entries of $\mathbf{x}_j$, the L-dimensional vector representation of document j.

The term frequency must be approximated in the case of streaming documents since the corpus is not fixed. Reference [13] states that functions different from TFIDF are needed when the term frequencies are not available from the start and points to approximations of TFIDF as in Reference [5]. Reference [5] addresses the problem of on-line document processing using functions of term counts but does not weight them by document frequencies. In doing so, the power of the TFIDF representation – to down-weight content-free words that appear in most of the documents – is lost.

There have been several algorithms developed for dynamic text clustering that ignore the effects of a changing corpus on the term weighting ([3], [7], and [14]). Reference [3] demonstrates a method termed complexity pursuit to extract topics of chat line discussions. Although topic-specific words are extracted, the process operates on vectors computed through an LSI computed on the full data matrix within a window. This is not a true streaming algorithm since the process must wait until the window is full. However, the approach could easily be modified to a fully streaming algorithm using methods similar to ours. Hung and Wermter's dynamic adaptive self-organizing model (DASH) [7] transforms the data into a TFIDF vector space representation before proceeding to treat the data as non-stationary. Zhong's on-line spherical k-means (OSKM) algorithm [14] is an adaptive clustering technique but the experimental results are collected on data that has been pre-processed as a static corpus. The data is pre-processed using the Bow toolkit and words that occur in less than three documents are removed from the lexicon. In a true streaming application, this would not be possible without windowing the data stream.

The above mentioned algorithms use the precomputed vectors: in order to obtain those vectors, the entire corpus, or a windowed subset of documents, must be available. The work addresses streaming text but uses a representation that cannot be achieved in a streaming context (the work develops streaming *algorithms* but applies them to precomputed text vectors). This is not a critique of the algorithms but an inherent property of the text mining approach that the vectors cannot be produced without the full corpus unless one makes an approximation to TFIDF. The method we suggest next could easily be incorporated into these algorithms.

B. A streaming approximation

One solution is to use a time window and calculate the document frequency within that window. In this work, we use an exponentially weighted moving average with a parameter α which allows for varying the amount of history influencing the value.

Instead of calculating the inverse document frequency, consider the document frequency, DF_j, for word j. That is,

$$DF_j = \frac{b_j}{D}.$$

Suppose that at each time, t, a new document is observed. Let $Y_j(t) \in \{0, 1\}$ indicate whether word j occurred in the document read at time t. Then we can approximate the document frequency for word j at time t by

$$DF_j(t) = \alpha Y_j(t) + (1 - \alpha) DF_j(t - 1) \tag{2}$$

which implements an exponential window on the documents. Documents can then be written as TFIDF vectors (as in (1)) by using the log of the inverse of (2).

C. Document similarity

Typically, document similarity is determined by measuring the cosine of the angle between the vector representations of the documents [12]. Since under the TFIDF representation the vector entries are in $\mathbb{R}^+$, the cosine of the angle between any two documents will be in $[0, 1]$. Documents that lie close to each other in document space will have a similarity measure close to one. Let $\mathbf{x}_a$ and $\mathbf{x}_b$ be the vector representations of documents a and b. Then

$$s(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a \cdot \mathbf{x}_b}{|\mathbf{x}_a||\mathbf{x}_b|} \tag{3}$$

is used as a measure of similarity between the documents.

III. A Graph Representation

We will represent the documents within the framework of a graph. The vertices (nodes) of the graph will contain statistics on a collection of articles that share a common theme or topic. The edges will represent similarities among the nodes. This representation is updated as each document is processed. The document is compared with the current nodes and is either added to a node or used to create a new node. The edges from the updated node are then updated to capture the changes to the topic within the graph. This representation performs two basic functions: it provides a reduced representation of the documents which incorporates information about the document topics and the relationships between topics and it provides a method for visualization and analysis of the topics and relationships. We will illustrate these ideas in Sections V and discuss possible extensions and future work in Sections VI. The graph is dynamic: the nodes and edges are created/deleted and modified as new documents are processed. Each node retains its own lexicon vector corresponding to the word frequencies for the documents in the node. This frequency vector is updated in much the same manner as (2) and is explained in Section III-A. The frequency vector (and other statistics taken from the documents, such as keywords, title words etc.) can be used to provide a summary of the node, the "topic" describing the node. These vectors are then used to define the similarity between nodes, and the edges are weighted by these similarities.

A. Node creation and updating

An upper limit is placed on the number of nodes in the graph. As documents are read, they can either be assigned to an empty node in the graph or used to update an existing node. Associated with each node is a list of words contained in the articles that have been assigned to the node. There is also a count for each of those words that is determined by the word occurrences in the assigned articles. The count is updated in much the same way as the document frequency, by using an exponential window.

Suppose a new article is read, and the length of the lexicon (including all words in the article) is L. Let $\mathbf{z}_a = (z_{a_1}, \ldots, z_{a_L})$ be the count vector for the words in the article. If the article is assigned to an empty node in the graph, then that node will be assigned the counts, $\mathbf{z}_a$.

If the article is assigned to a node that is not empty, the counts for the node will be updated. Suppose the article is assigned to node $\mathfrak{N}$. Let $\mathbf{z}_{\mathfrak{N}}$ be the count vector for node $\mathfrak{N}$ prior to the assignment. That is, $\mathbf{z}_{\mathfrak{N}} = (z_{\mathfrak{N}_1}, \ldots, z_{\mathfrak{N}_L})$, where the vector is zero-padded to length L if necessary. Then the entries of $\mathbf{z}_{\mathfrak{N}}$ will be updated by

$$z_{\mathfrak{N}_k} \leftarrow \begin{cases} z_{a_k} & \text{if } z_{\mathfrak{N}_k} = 0 \\ \beta z_{a_k} + (1-\beta) z_{\mathfrak{N}_k} & \text{otherwise} \end{cases} . \tag{4}$$

As with the document frequencies for the words in the lexicon, this implements an exponentially windowed average for the counts in the node. This has the effect of reducing the contribution of the words from older articles. Reference [14] also uses an exponential decay term in the OSKM algorithm and points out the intuitive fact that "one needs to forget [in order] to be adaptive".

The $\mathbf{z}_{\mathfrak{N}_1}, \ldots, \mathbf{z}_{\mathfrak{N}_N}$ contain the counts for the N nodes in the graph. To obtain the vector space representation of the nodes, these counts are multiplied by the log of the approximation to the inverse document frequencies as in (1). When a new article is read, the similarity between the article and all existing nodes in the graph is calculated using (3). If the most similar node has $s \geq \tau$, the article is added to that node according to (4). If the maximum similarity is less than τ and there are empty nodes in the graph, the article is put in an empty node. If the maximum similarity is less than τ and there are no remaining empty nodes, the article is put in the node that has gone the longest without being updated. In that case, the counts for the node are reset to the counts for the article. (Note that this is equivalent to setting $\beta = 1$ in (4).)

The update rule in (4) requires the parameter β. One method for adaptively choosing β is to set

$$\beta = \begin{cases} 1 - s & \text{if } s \geq \tau \\ 1 & \text{if } s < \tau \end{cases} . \tag{5}$$

This has the effect of making large (small) changes to the node when the similarity between the article and the node is small (large).

To see what this rule does, consider the extreme cases. If the match to the node is perfect ($s = 1.0$) then the new document is providing no new information, so it is reasonable to leave the node unchanged; if $s < \tau$ for all nodes, then either a new node needs to be added (if the graph has not yet reached its maximum order) or an old node needs to be replaced with the new document, starting a new topic; if $s = \tau$ then the document is very far from the topic of the node, indicating that the topic has changed, and the node statistics need to be changed maximally to adapt to the change. The parameter τ controls the amount a topic can change before a node is reset.

B. The adjacency matrix

In addition to the statistics from the news articles that are associated with each node, the graph adjacency matrix is used to retain information about the connections between the news topics. For a graph of order N (the order is the number of

vertices in the graph), the graph adjacency matrix, $\mathbf{A}$ is an $N \times N$ matrix where A_{ij} holds information about the edge from vertex i to vertex j. In this work, the graph is undirected so the adjacency matrix is symmetric. For an unweighted graph the adjacency matrix is a binary matrix with $A_{ij} = 1$ if there is an edge from vertex i to vertex j and $A_{ij} = 0$ otherwise. The graph edges can also be weighted so that $A_{ij} \in \mathbb{R}$. In this work, the graph is weighted and the adjacency matrix holds the similarities between the vertices where the similarity is calculated using (3). Each time a node is updated, the similarity between that node and all other nodes in the graph is updated. That is, if vertex i is updated, then the i^{th} row and column (which are identical in the symmetric matrix) are also updated. This updating scheme is an approximation. Since the lexicon weights have changed one would need to update the entire adjacency matrix to reflect the change. In order to reduce computation, and thus allow larger graphs (more topics), we chose to implement this approximation. The approximation may be skipped if the graph is sufficiently small and/or the rate of the stream is slow enough to allow full updating.

The adjacency matrix can be used to combine nodes that are similar. This can be done either by considering just the between-node similarity, or by using the neighborhood information within the graph to combine nodes that are both highly similar and have the same set of "most similar" neighbors.

IV. The Data - Google News

News articles were collected every day from January through August of 2006. Articles were retrieved through links provided by the Google news website using five categories defined by Google. The news categories and their labels were World (W), US (U), Business (B), Health (H), and Science and Technology (S). There were approximately 300 articles collected per day.

The articles were stripped of HTML tags and parsed in order to find the body of the article. The articles contained advertisements and links to other articles so effective parsing was important in order to ensure that the text clustering methods were working on the article body and not on other words that were unrelated to the articles. Perfect parsing was not possible and in cases where articles were found to be parsed incorrectly, they were eliminated from the data set. Still, it is possible that some retained articles were incorrectly parsed.

Articles were also stripped of non-alphabetic characters (all numbers and punctuation), hyphenated words were split into separate words, and words were stemmed. The stemming was performed using the Porter stemming algorithm [10]. Finally, the remaining words with two or more letters were counted for each document.

The word counts from the cleaned, parsed, and stemmed article were used to create a vector representation for each article. In order to approximate the TFIDF representation, term frequencies for each article were multiplied by the $\log$ of the approximation of the inverse document frequency for each word as in (1). The document frequency for each word was calculated by (2). Each new article necessitated changing the document frequency for every word.

A growing lexicon was used such that new words were added to the end of the lexicon. In this preliminary work, there was no limit placed on the size of the lexicon. It is often the case that a list of context-free words, called a stop list, is used to eliminate the consideration of certain words in the document representation [9]. We did not employ a stop list in this work. It was expected that the low weighting from the TFIDF representation would discount any effect from such words and the savings of only a few hundred words was not a consideration.

In order to illustrate the process, Fig. 1 shows a small graph of ten nodes. Articles from the categories of Science and Technology and World news were processed from January 1 through January 12. The titles displayed are those from the articles assigned to each node since the last time the node was reset. Since the graph is so small, nodes are reset often. For larger graphs, articles are retained in the nodes longer and there is more clustering of articles from similar topics. In this graph, we see three nodes that contain more than one article: a node about the disappearance of frogs due to global warming, a node about astronomy, and a node about Apple™.

The gray-scale of the edges is proportional to the cosine similarity between the nodes. Most of the similarities are small because articles were placed within nodes to which they were similar or else they were placed in their own node. The similarity between all nodes is smaller than the similarity threshold τ. Still, the largest of these similarities are seen in the edge between the nodes about Apple and North Korea and the edge between the nodes about stem cells and frogs. The stem cell and frog pair shares the word "research" which is weighted heavily in the TFIDF representation and therefore contributes to the cosine similarity. The North Korea and Apple pair shares the word "China" due to a problem parsing the article "Byte of the Apple". There are links at the bottom of the Apple article to articles about news in China. This contributes to the cosine similarity since the articles do share high weight words but this is due to the imperfect parsing rather than to any connection between the stories.

V. Visualization

The graph was built using an adaptive value of β as in (5). The document frequency was approximated using $\alpha = 0.1$ in (2). The maximum order of the graph was set to 200. The similarity threshold in (5) was $\tau = 0.1$. The data were articles from January 2006 from the five classes described in Section IV. Fig. 2 shows the articles assigned to one node in the graph over several days' time. The first article is about bird flu. There were no further bird flu articles, so this node remained unchanged while other nodes were updated, until it became the oldest node in the graph. The article about Symantec was not similar (above the threshold $\tau = 0.1$) for any nodes, and so was assigned to this, the oldest node. Once this article is assigned to the node, the node is re-initialized and the words and corresponding weights from the first article are

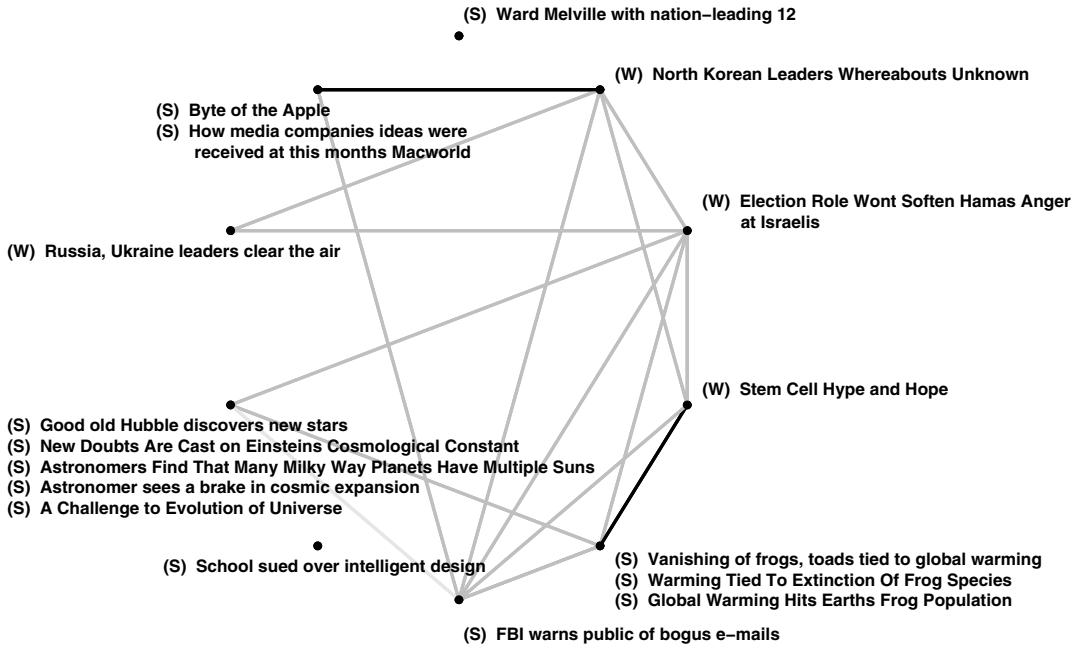

Fig. 1. **A small example graph** – The titles displayed are those from the articles assigned to each node since the last time the node was reset. The gray-scale of the edges is proportional to the similarity between the nodes.

dropped. The following two articles are also about Symantec and are assigned to the node due to the high similarity. The fourth article, "Ahern leads trade mission to India", has low similarity but is assigned to the node due to the node remaining unchanged longer than any other node in the graph. All of the articles that follow are assigned to the node due to their similarity and show the evolution of the meaning of the node. We see that the node evolves from *India* to *India and Pakistan* to *Pakistan and the United States*, to *terrorism* and finally to *al-Qaida*. The latter articles show variations of the spelling "al-Qaida" and "al-Qaeda".

Fig. 3 gives another view of this node. The axes correspond to the words in the node and the article titles (time) with gray-scale value depicting the weight of the word at the time the article is incorporated into the node. For each article, the words chosen to display are those with the three highest TFIDF values. In addition to the top three words, the TFIDF value for other words is shown due to overlap in the lexicon at different times. For example, the Symantec articles also have the word "attack" which is a high weight word from the eleventh article ("US: Contacts with Pakistan Positive Despite Attack Protests").

The evolution of the word importance can be seen in the diagonal gray region. Early on, the important words are "Turkey", "bird" and "flu". Then words about Symantec become important for a while. The two resets are clear in this picture, indicated by the blocks in the lower corner. Then words about India become important, followed by Pakistan and then words related to terrorism. We can see here the progression of the topic about the India/Pakistani subcontinent morphing from stories about trade missions into stories about terrorism. The rate at which the node is allowed to change is driven by the parameters α and τ, which control the memory of the lexicon and the threshold of similarity for addition to the node.

VI. Extensions

There are several obvious extensions that could be incorporated. First, the nodes could be allowed to split. For instance, in the above example if articles about trade missions continued to be observed while articles about terrorism were also observed, one might want to split this node into two. This can be implemented through a hierarchical graph method, allowing each node to spawn its own subgraph. Thus, we would have high level generic topics, with subtopics in a graph underneath each topic.

We currently assign each article to a single node. It would be easy to allow articles to be assigned to multiple nodes,

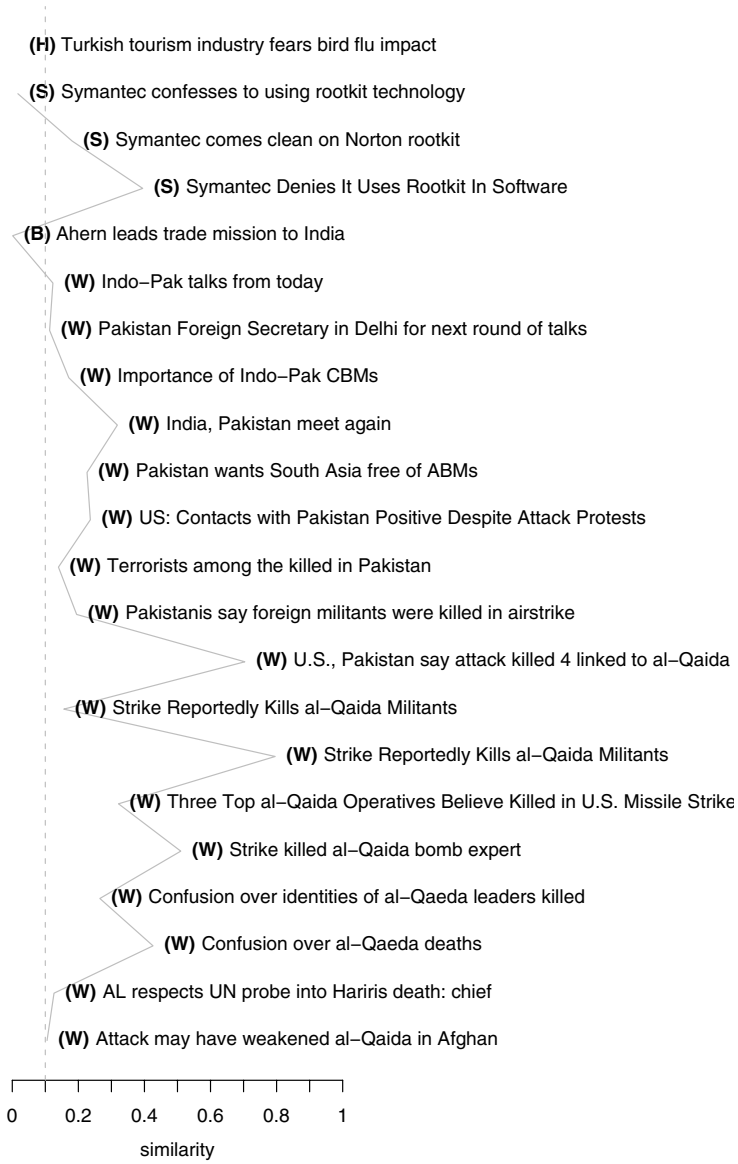

Fig. 2. **Articles assigned to one of the nodes in the graph** – The first article is shown at the top, and the titles of the added articles are listed in order. The similarity between the article and the graph node is shown by the gray line to the left of the article titles. The value of τ is 0.1, and articles with a similarity less than this value were added to the node not due to similarity but because the node had been unchanged for longer than any other node in the graph.

Fig. 3. **Important words in the node over time** – The words chosen for display are those with the three highest TFIDF values after the addition of each article. The gray-scale is proportional to the TFIDF value.

proportional to their similarity. However, care must be taken to ensure that this doesn't cause the nodes to all move together, resulting in a set of essentially identical nodes, each about "everything" and none specifically about anything.

Both α and τ are static in this implementation, parameters set by the user. We are investigating a method for adapting α by monitoring the "stable" words in the lexicon. That is, an appropriate value of α will result in a near-constant document frequency for some words. By adapting α based on the document frequency of these words, different text sources will adapt at different rates. For example, on-line news may adapt more slowly than a set of web logs but more quickly than articles from an on-line science journal. An adaptive and node-specific value of τ is another area of research and would allow different nodes in the graph to change at different rates.

VII. Discussion

We have described a method for analysis of streaming documents that uses dynamic lexicons and word weighting to organize the topics into a graph structure. We have not discussed the graph itself in depth, focusing in this paper on the methods for populating the nodes of the graph. Future work will involve extracting useful information from the graph, and implementing various extensions of these ideas.

We have not presented an evaluation of our methods. We have used the graph on benchmark datasets where time can be extracted such as the Yahoo! 20 newsgroups dataset [1]. However, the graph is not meant to perform text categorization so even testing class purity within the nodes is difficult since the nodes are meant to evolve in time. We are currently testing the TFIDF approximation in benchmark datasets using a simple classifier and measuring the change in classification rate.

The graph provides us with a convenient structure within which to both analyze the current topics and merge topic threads when they become similar. A hierarchical version of the graph would allow splitting of topics into subtopics. Extracting invariants from the graph may be a problem in a streaming environment, since many of the things one would like to do (spectral clustering, for example) are quite computational in nature. Fast approximations are needed to provide analysis in a streaming fashion. More sophisticated analysis could be done off-line, to provide snapshots of the document stream.

We have shown only the most rudimentary of visualization methods in this paper. More sophisticated ways of visualizing the data are clearly needed. Relationships between topics can be investigated through the similarities or links in the graph and descriptions of topics can be improved through dynamic lexicon weighting. Better ways to describe the topics and their relationships is an area for future research.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html

Acknowledgment

This work was funded by the Office of Naval Research through the In-House Laboratory Independent Research (ILIR) program.

References

[1] J. Allan, *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA: Kluwer Academic Publishers, 2002.

[2] J. Allan, "Introduction to Topic Detection and Tracking", in *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA: Kluwer Academic Publishers, 2002.

[3] E. Bingham, A. Kaban, and M. Girolami, "Finding topics in dynamical text: application to chat line discussions," in *10th Int. World Wide Web Conf. Poster Proc.*, 2001, pp. 198-199.

[4] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey, "Scatter/gather: A cluster-based approach to browsing large document collections," in *Proceedings of ACM SIGIR*, 1992, pp. 318-329.

[5] I. Dagan, Y. Karov and D. Roth, "Mistake-Driven Learning in text Categorization," in *Proceedings of the Second Conference on Empirical Methods in NLP*, 1997, pp 55-63.

[6] I.S. Dhillon and D.S. Modha. "A data clustering algorithm on distributed memory machines," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[7] C. Hung and S. Wermter, "A dynamic adaptive self-organizing hybrid model for text clustering," in *Proceedings of the 3rd International Conference on Data Mining (ICDM 2003)*. Melbourne, FL: IEEE Computer Society, December 2003, pp. 75-82.

[8] X. Lin, D. Soergel, and G. Marchionini, "A self-organizing semantic map for information retrieval," in *Proceedings of the 14th Annual international ACM SIGIR Conference on Research and Development in information Retrieval*, New York: ACM Press, 1991, pp. 262-269.

[9] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, Cambridge, MA: MIT Press, May 1999.

[10] M.F. Porter, "An algorithm for suffix stripping," *Program*, 14(3), 1980, pp 130-137.

[11] E. Rennison, "Galaxy of News: An Approach to Visualizing and Understanding Expansive News Landscapes," in *UIST 94, ACM Symposium on User Interface Software and Technology*, New York: ACM Press, 1994.

[12] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, New York: McGraw-Hill, 1983.

[13] F. Sebastiani, "Machine learning in automated text categorization," Tech. Rep. IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.

[14] S. Zhong, "Efficient streaming text clustering," Neural Networks, 18(5-6), 2005, pp 790-798.