A Graph Analytical Approach for Fast Topic Detection

Hassan Sayyadi, University of Maryland
Louiqa Rschid, University of Maryland

Scalability and accuracy challenges when processing large noisy data collections such as social media require new solutions for Topic Detection. We present KeyGraph, an efficient method that improves on the current topic detection methods by considering keyword co-occurrence. We show that KeyGraph has similar accuracy when compared to state-of-the-art approaches on small well annotated collections. Further, KeyGraph can successfully filter noise and identify events in social media collections. An extensive evaluation using mechanical Turk demonstrated both the accuracy and recall of KeyGraph. The runtime performance of KeyGraph is significantly better when compare to LDA topic models on large collections.

Categories and Subject Descriptors: H.3.3 [**Clustering**]: Information Search and Retrieval; E.1 [**Graphs and Networks**]: Data Structure; H.2.8 [**Data Mining**]: Database Applications

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Topic Detection, Network Analysis, Community Detection, KeyGraph based Topic Detection

1. INTRODUCTION

A topic (event) is defined in the Topic Detection and Tracking (TDT) community as a specific event or occurrence at a specific time and in a specific place [3]. New Event Detection (NED) and Retrospective Event Detection (RED) are two groups of event detection algorithms [30].

There have been many successful TDT approaches reported in the literature [5; 13; 30]. Initial methods typically relied on clustering documents. Consider a core group of keywords representing a specific topic. Documents that contain these keywords will be more similar to each other based on a metric such as TF/IDF[23]. Thus, variations of TF/IDF were used to compute keyword based feature values, and cosine similarly is used as a similarity (or distance) measure to cluster documents.

The next generation of TDT approaches moved the analysis from document clustering to keyword based features. Recent research has addressed the relationships between keywords, their occurrence in documents, as well as the dynamics of such relationships [4; 10; 19; 24; 29]. A variety of machine learning techniques have been utilized. Topic modeling approaches based on latent Dirichlet allocation (LDA) [7; 26] exemplify these advances in TDT.

The gold standard algorithms that are either based on document clustering or on keyword based machine learning have worked well for many collections. However, the rapid growth of social media, and the corresponding need to handle large document collections in a noisy environment, has created a need for scalable and accurate TDT approaches. Social media collections differ from collections of articles along several metrics including the size distribution of documents and the distribution of unique keywords in the collections. The challenge is to rapidly filter noise or irrelevant documents while at the same time being accurate when clustering a large collection of documents.

In this paper, we present KeyGraph, a novel and efficient approach for topic detection that is inspired by the keyword co-occurrence graph and efficient network analysis methods. We explain the intuition for KeyGraph and keyword co-occurrence by citing the research of Newman et al. [16]. They performed an evaluation of topic coherence across many methods and metrics and empirically demonstrated that keyword co-occurrence, computed via simple pointwise mutual information (PMI), was the most consistent performer, achieving the best or near-best results over multiple datasets. In some cases the results approached or surpassed human inter-annotator agreement.

Current topic modeling methods focus on word distributions in the document and they do not explicitly consider word co-occurrence. Extending topic modeling to include co-occurrence is a computationally daunting challenge. KeyGraph is a first approximation; it represents a collection of documents as a keyword co-occurrence graph. It then applies an off-the shelf community detection algorithm to group highly co-occurring keywords into communities. Each community forms a constellation of keywords and corresponds to a set of topic features that represents a topic. Our intuition, which we prove empirically, is that communities in the KeyGraph are a good proxy for topics.

We compare KeyGraph with other solutions on the well known TDT4 dataset. We further compare KeyGraph with LDA topic modeling using Gibbs Sampling (LDA-GS) [2] on a large social media dataset. We perform a human evaluation of the accuracy of topic detection. Our research makes the following contributions:

- We demonstrate that the accuracy of the KeyGraph method for topic detection is similar to the gold standard algorithms, including kNN [5], GAC [30], two variants of LDA topic modeling [1; 2] and a probabilistic model [13]), on the well known **TDT4** dataset.
- Using the *Spinn3r* data collection, a large noisy collection of blog posts, we demonstrate that KeyGraph can filter noisy irrelevant posts and create smaller clusters of relevant documents for each topic. In contrast, a topic modeling solution is not able to filter as many irrelevant documents.
  A human evaluation using Amazon's Mechanical Turk marketplace is used for validation of the detected events. The best topics or events chosen by KeyGraph and LDA topic modeling with Gibbs Sampling (LDA-GS) were evaluated by humans. On average, only 40% of the documents selected by LDA-GS received votes of relevant or highly relevant. In contrast, over 77% of the documents selected by KeyGraph received votes of relevant or highly relevant. Both methods were shown to have high recall using a *pseudo-recall* metric. There was high inter-annotator agreement on the ratings.
- We demonstrate that KeyGraph (whose running time is linear in the size of the collection) outperforms LDA-GS [2] by an order of magnitude. We use a subset of up to 113,000 documents from the *Spinn3r* data collection and show that the performance improvement is increasingly significant as the number of documents exceeds 50,000 documents.

This paper is organized as follows: In the next section, we review related work. Section 3 presents the KeyGraph topic detection algorithm. Next, in Section 4, we compare the accuracy of KeyGraph to other gold standard solutions on the TDT4 dataset. We also compare KeyGraph and LDA topic modeling on the *Spinn3r* data collection.

2. RELATED WORK

Event or topic detection problems usually fall in two categories, namely New Event Detection (NED) and Retrospective Event Detection (RED). While NED algorithms deal with documents processed online, RED is defined as the offline discovery of events in a historical corpus [30]. The solution approaches can be classified as document based clustering and keyword based.

NED models are proposed as a single pass incremental clustering algorithm. For a newly arrived document, the similarity between the document and known events is computed, and the maximum similarity will be determined. If the similarity is more than some a priori threshold, then the document will be assigned to the corresponding event. Otherwise it will be considered a new event. Allen et al. [5] used kNN (k-Nearest Neighbors) with a variation of *TF/IDF*. They also penalized the threshold based on the elapsed time interval between the document and the event; the threshold increases with this elapsed time interval. Since future document features are not known a priori, such online clustering algorithms need to estimate *IDF*. While Allen et al. [5] use an auxiliary dataset to estimate *IDF*, Yang et al. [30] proposed an *incremental IDF* factor which will be updated for each new incoming document. Brants [8] introduced incremental TF/IDF and similarity score normalization for NED. Kumaran et al. [12] improved their results through the use of text classification techniques as well as by using named entities.

Retrospective Event Detection (RED) was originally defined by Yang et al. [30]. Many text clustering algorithms have been developed for RED. One of the most accurate is by Yang et al. who proposed an agglomerative clustering algorithm, GAC (augmented Group Average Clustering), to cluster documents in the corpus. They also applied an iterative bucketing and re-clustering model that was proposed by Cutting et al. [9] to control the tradeoff between cluster quality and computational efficiency. Similar to NED, RED models also use clustering algorithms, but they offer different variations for document representation, and for distance or similarity metrics. For example, [25; 28] have shown that a set of significant words, including named entities, can be used for efficient search result clustering. Mory et al. [14; 15] also propose a new document representation model in which they extracted the maximally connected components from the word co-occurrence graph [18]. These components were called foundations or basic concepts, and are used as document features. Documents represented with basic concepts were clustered using a *single-link* clustering algorithm.

Next, we discuss another approach to event detection that will focus on words/keywords and their correlation. He et al. [10] study feature trends that may vary over time and they created and analyzed time-series word signals. Next, they chose correlated word signals as features for topic detection. Similarly, Prabowo et al. [19] built word co-occurrence graphs and looked at the graph community structure for monitoring and tracking debates in online communities.

Based on [7; 26], the Latent Dirichlet Allocation (LDA) topic model is a three-level hierarchical Bayesian model. Each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. LDA has been widely used and has been shown to be a successful algorithm for topic modeling. The Mallet toolkit provides an implementation of LDA which uses Gibbs sampling as the inference engine(**LDA-GS**)[2]. Blei also provides another variant of LDA which uses variational expectation maximization (VEM) for latent Dirichlet allocation (**LDA-VEM**)[1].

Li et al. [13] proposed a novel solution that included a probabilistic algorithm which directly exploits time for topic modeling. Documents and events are represented by features including *persons, locations, keywords*, and *time*. The probability of each document is defined as:

$$\begin{aligned} p(article) &= p(time) * p(persons) * p(locations) \\ &\quad * p(keywords) \\ p(x_i|e_j) &= p(time_i|e_j) * p(persons_i|e_j) * p(locations_i|e_j) \\ &\quad * p(keywords_i|e_j) \end{aligned}$$

where x_i and e_j are ith article and jth event. They computed an estimation of the event count using the time distribution of the article count. Next, an Expectation Maximization (EM) algorithm is used to maximize the log-likelihood of the distributions of these features and to learn the model parameters. Such an algorithm requires the number of events to be given a priori which makes this approach somewhat impractical and difficult to tune.

3. KEYGRAPH BASED TOPIC DETECTION

Topic modeling represents topics as an infinite mixture over the probability distributions of sets of words. Our intuition is to consider word co-occurrence as well. Ideally, we would extend the framework so that a topic is an infinite mixture over the probability distributions of sets of words *such that the probability distribution of word co-occurrences among these sets of words is maximized*. We note that this extension can no longer be addressed by the current 3 level hierarchical modeling approach of topic modeling. Further, the computational complexity of such an extension which may have to simultaneously consider the co-occurrences of subsets of keywords is not known. Thus, as a first approximation to this extension, KeyGraph will focus on the challenge of maximizing word co-occurrence *alone*. The problem then becomes almost identical to community detection over the keyword co-occurrence KeyGraph. KeyGraph, presented in Algorithm 1, comprises the following phases:

(1) Building the KeyGraph.
(2) Extracting topic features.
(3) Assigning topics to documents.

In the first phase, we construct a keyword co-occurrence graph labeled a KeyGraph. A KeyGraph has one node for each keyword in the corpus. Edges in a KeyGraph represent the co-occurrence of the corresponding keywords and are weighted by the count of the co-occurrences.

The second phase of the algorithm includes community detection and feature extraction, i.e., a constellation of keywords to represent a topic. From Figure 1, constellations or sets of keywords which capture some meaningful topic are more densely linked, while there are fewer links between keywords that participate in different communities. Community detection approaches apply relational clustering algorithms from network analysis. Several algorithms for community detection on static graphs are presented in [17]. We use an off-the shelf algorithm based on the *betweenness centrality* metric that has been shown to perform well.

The third phase of the algorithm is to use the features to assign topics to documents. Finally, for each pair of topics, if there are multiple documents that are assigned to both topics, then we assume that these are sub-topics of the same parent topic, and we merge the sub-topics.

Recall that Li et al. [13] incorporated time into topic modeling. The KeyGraph is dynamic since the documents are time stamped. Community detection for dynamic networks is an emerging problem. While some solutions are presented in [21; 22], their accuracy has not been well studied. The complexity and cost of dynamic community detection is also much greater compared to solutions for static networks. Therefore, in our current solution, we aggregate keyword co-occurrences over time. This allows us to use an off-the-shelf static community detection algorithm. We further note that our first priority is to extend topic modeling to consider both word and keyword co-occurrence distributions. We feel this extension has greater potential compared to temporal features, for noisy social media.

Preliminary results of the KeyGraph approach to topic detection were presented in [24]. Our prior work has been extended to include a comparison against the gold standard approaches based on accuracy. We also extended the prior research with an extensive human evaluation and scalability experiments.

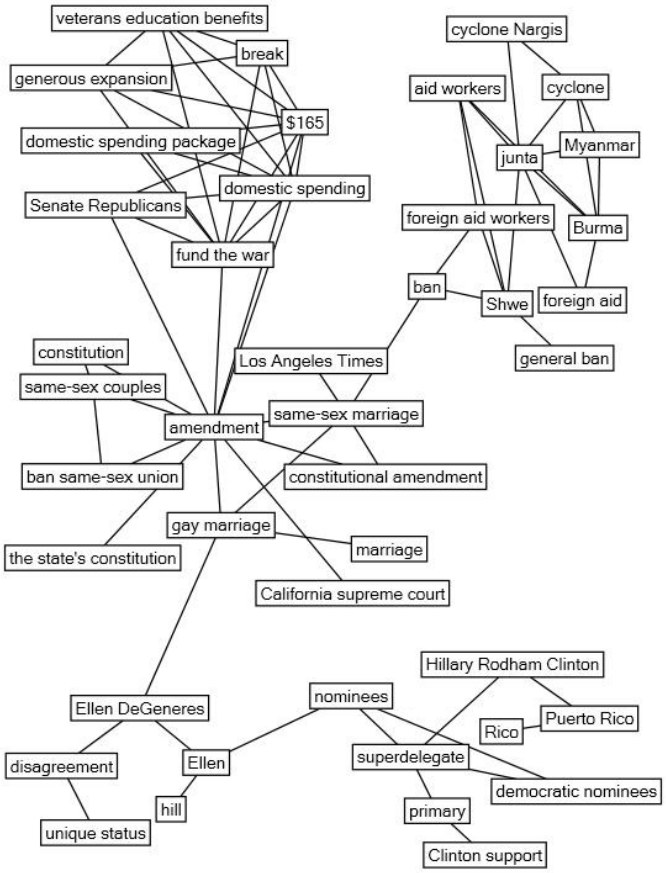

Fig. 1. A small sample of the KeyGraph

Building the KeyGraph: We first extract keywords from each document and create one node per unique keyword in the dataset. Keywords can include the terms, noun phrases, and named entities in the document. An edge $e_{i,j}$ between nodes n_i and n_j is added if keywords w_i and w_j co-occur in at least one document.

After building the initial co-occurrence graph, we calculate the document frequency (DF_i) for each node n_i. Nodes with a document frequency below a pre-specified threshold are removed. Similarly, we remove all edges $e_{i,j}$ where the keywords w_i and w_j co-occur in at most one document. We then compute the conditional probabilities $p(k_i|k_j)$ and $p(k_j|k_i)$ for each edge $e_{i,j}$. The conditional probability is computed as follows: $P(k_i|k_j) = \frac{DF_{i\cap j}}{DF_j} = \frac{DF_{e_{i,j}}}{DF_j}$. The edges with both conditional probabilities below a pre-specified threshold are also removed.

Extracting Topic Features: Our intuition is that keywords co-occur when there is a meaningful topical relationship between them. Based on this analogy, we model the KeyGraph as a social network of relationships between keywords. The occurrence of an event will change the community structure of the KeyGraph. The number of co-occurrences of the topic keyword features will increase due to the increase in the

Algorithm 1 Graph Analytical approach for Topic Detection

```
for all Document $d_i$ do
    Extract words, noun phrases and named entities as document keywords/features
end for
%Building the KeyGraph
Build the keyword co-occurrence graph
%Extracting Topic Features
Extract strongly connected communities of keywords in the KeyGraph as topic features
%Assigning Topics to Documents
for all Extracted topic feature $f_t$ do
    Find the likelihood of topic $t$ for document $d$ as follow:
        $p(t|d) = \frac{cosine(d, f_t)}{\sum_{t \in T} cosine(d, f_t)}$
end for
for all Topic $t_i$ and $t_j$ do
    if the overlap of $t_i$ and $t_j$(between their document set) is greater than $\mu$ then
        Merge $t_i$ and $t_j$
    end if
end for
```

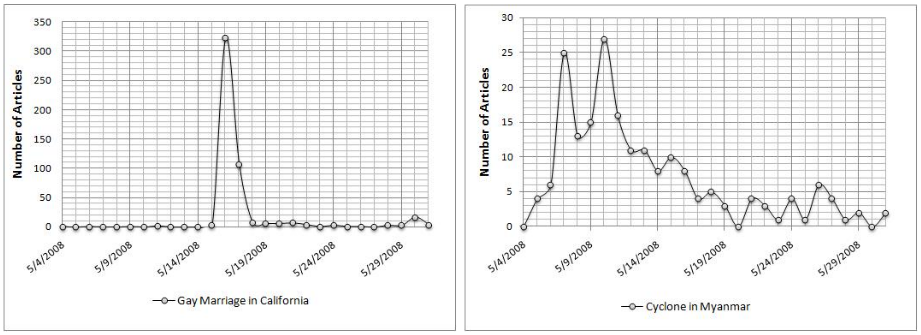

Fig. 2. Document frequency of topics *Litigating gay marriage in California* and *Cyclone in Myanmar*

number of documents related to the topic, sharply at the beginning or onset of the event. Then, depending on the nature of the event, the co-occurrences will decrease slowly or more rapidly. The corresponding number of relevant documents that will be grouped with this event will reflect a similar pattern. Figure 2 shows the document frequency for the two biggest events we found in the *Spinn3r* dataset (see Table I for the details of *Spinn3r* dataset). Both diagrams clearly show the general document frequency pattern of the topics.

For example, there are several communities of keywords in Figure 1 which are suggestive of specific topics. Within each community, the nodes are highly connected while connections to nodes outside of the community are rare. For example, the group of keywords including ”*cyclone*”, ”*Myanmar*”, ”*foreign aid workers*”, etc. creates a community to represent the topic *cyclone in Myanmar*. This broader topic includes the more specific topic that occurred a few days after the onset of the cyclone, when the Myanmar government denied access to foreign aid workers and refused to accept international aid. We note that the more general and more specific topic correspond to the two peaks in Figure 2. Another example which can be seen, describes *the amendment to the constitution to ban same sex unions in California*. A community of keywords including "*Same-sex marriage*", "*California supreme court*", etc. is visible in the graph."*The Senate approval of $165 billion to fund the wars in Iraq and Afghanistan*" and "*Ellen DeGeneres's opposition to the gay marriage*" are other topics which can be detected by the communities of keywords in this graph.

The betweenness centrality score is a good measure to find the edges between communities in a network [17]. Betweenness centrality for an edge is defined as the count of the shortest paths, for all pairs of nodes in the network, that pass through that edge. Inter-community edges will always obtain a high score, since the shortest paths between nodes from different communities will always pass through these edges. For instance, in Figure 1 the edges ("Ellen DeGeneres", "gay marriage"),("Ellen", "nominees"), and ("ban", "same sex marriage") are among the edges with the highest betweenness centrality score.

We use the following procedure to remove inter-community edges. The set of shortest paths between all pairs of nodes is calculated using a breadth first search (BFS). Then, the edge with the highest betweenness centrality score is selected to be removed from the graph. If two edges have the same betweenness centrality score, the one with the lower conditional probability value will be selected. This process will be repeated until no edge with a score greater than a pre-specified threshold can be found, then we can say we removed all inter-community edges from the network. Next, if the KeyGraph is not fully connected anymore, we will continue this operation for each smaller connected component recursively until all edges in the KeyGraph have a small betweenness centrality score. Stop conditions will be discussed in detail in section 4.5.

After all inter-community edges have been removed, each connected component of the KeyGraph represents a hypothesis about an topic. The highly co-occurring keywords in each component form the features for the topic.

As a natural consequence of aggregating the co-occurrence graph over time, the number of nodes will not increase but the density of the graph will increase [19].It is obvious that the new keywords that can be added to the graph is limited but keywords that never appear together in the past can appear in the same documents for new topics which makes the graph denser. We note that while the community structure of the KeyGraph changes over time, our approach aggregates keyword co-occurrences. Although such aggregation has the potential to miss the true community structure, our experimental results show that the quality of topic detection with our aggregated KeyGraph is as good as the best topic detection models in the literature.

Assigning Topics to Documents: After finding the communities of significantly co-occurring keywords as topic features, the next step is assigning the topic likelihoods to each document. In previous step, we extracted the keyword community from the KeyGraph. Each community of keywords represents a synthetic document, called a *feature document*, for the corresponding topic. The likelihood of topic t for a document d is determined by the cosine similarity of d and the corresponding *feature document* f_t.

$$p(t|d) = \frac{cosine(d, f_t)}{\sum_{t \in T} cosine(d, f_t)}$$

Table I. Details of Dataset 1 (part of the TDT4 Dataset)

Time	Oct. 2000 - Jan. 2001
Number of Topics(Events)	71
Number of topic articles	1923
Number of unique articles	1884
Average articles per event	27

We use the following approach to determine the TF value of a keyword in a feature document. There is an agreement that the most common keywords for a topic should have the biggest weight. We expect that the more closely the keyword is representative of a topic, the more it occurs with other keywords on that topic. Hence, we use the average co-occurrence of each keyword from the *feature document* with other keywords of the *feature document* as TF value. We experimented with other options including using the equal values (TF=1), using the node degree as TF, etc, but they did not improve performance.

Since *documents are assigned a likelihood or probability distribution over topics*, our model may assign an article to multiple topics unless we explicitly place a restriction that a document can only be assigned to a single topic. This provides flexibility to our method to consider multi-topic documents. It also helps to overcome network community detection errors that may have occurred in an earlier step.

A common problem with any clustering or community detection algorithm is that a true cluster can be split. Our soft clustering capability which allows a document to be assigned to multiple topics can address this problem. Suppose that a KeyGraph community related to a topic is mistakenly split into two or more sub-communities. This results in having several sub-*feature documents* for the topic. After document assignment, there would be a significant overlap in the corresponding document sets. This occurs because the documents related to the topic contain words from all sub-*feature documents*. We allow our algorithm to partition the KeyGraph into a fairly large number of communities to make sure that we exceed the expected number of topics. We then merge those topics with a significant overlap in their relevant document sets. We demonstrate the robustness of our approach as we change multiple parameters in our experimental evaluation.

4. EVALUATION

We evaluate KeyGraph as well as several gold standard algorithms that have been reported in the literature. We show that KeyGraph has similar accuracy to the gold standard algorithms on the well known **TDT4** dataset; this is a small and well annotated dataset. We demonstrate further that KeyGraph can successfully filter noise and identify events in a social media collection. KeyGraph has good accuracy and recall. The running time of KeyGraph significantly outperforms LDA topic models on large collections.

4.1. Datasets

TDT4: The **TDT4** collection is a well known dataset that has been widely used to compare methods for topic detection [27; 13; 30]. The original data collection includes more than 15,000 documents. However, only a subset of it has been annotated by humans. Similar to all other methods, we use the annotated subset of collection for the purposes of evaluation.

The **TDT4** dataset includes 71 human identified topics together with 1923 documents. Due to document overlap of some topics, there are 1884 unique documents. We exclude topics that have less than 4 documents. Dataset statistics are in Table I

Table II. Details of Dataset 2 (subset of the Spinn3r dataset for human evaluation)

Time	May 1, 2008 - May 7, 2008
Number of topics	[unknown]
Number of sources	1905
Number of unique articles	32041
Average articles per source	1.6

Spinn3r: The *Spinn3r* collection includes approximately 44 million blog posts collected over 3 months in 2008. This includes millions of posts which are on a wide range of topics. The document size ranges from one sentence to very large documents. The posts originate from many thousands of sources, and no text extraction or data cleaning was performed. Thus, this represents a large and noisy collection. A random subset of the *Spinn3r* collection labeled Dataset 2 was used for human evaluation; details of Dataset 2 are in Table II. A larger subset of up to 113,000 documents was used to compare the running time of the methods.

Feature Extraction: For both datasets, we tokenize the text and extract all words from the documents. After removing stop words, we use a Porter Stemmer to stem the tokens. Next, we extract noun phrases and named entities since both are found to be informative and important for TDT tasks. We use the Stanford Parser for noun phrase extraction and IdentiFinder for named entity extraction. To extract named entities, we use IdentiFinder. Like other named entity recognition tools, IdentiFinder is sensitive to sentence structure and it is case sensitive. For example, the named entity ”obama” which starts with a small letter in the document, will very likely be missed. Some named entities also will be missed because of the sentence structure. To reduce such false negatives, we use a two pass approach. In the first pass, we run IdentiFinder on the text collection and collect all named entities recognized by IdentiFinder. We use this collection as a reference database. In the second pass, for each word in a document, we consult the reference database to determine if the word is a named entity. For example, if ”Obama” appears in at least one document in the corpus and is detected as a named entity by Identifinder, it will be added to the reference database. Next, in the second pass, we do a *case insensitive* search to find all occurrences of ”Obama” in the corpus. Hence, all occurrences of ”obama” or ”Obama” in corpus will be marked as named entities, independent of the letter case and the sentence structure. This way, we can retrieve some of the named entities which were initially not detected by IdentiFinder. However, words like ”House” will be a true positive named entity in some documents but they are false positives in other documents. Our two pass method may generate some false positives. Our experience is that the number of false positives generated in the second pass are negligible compared to the many thousands of false negatives that we avoid using the two pass approach. When processing each document, we boost the term frequency of keywords that are identified as named entities so that they are emphasized by our topic detection algorithm. We obtained the best results by doubling the term frequency of named entities.

Noise: The size (word count) of social media documents has high variance and there are many short (one sentence) documents. Simultaneously, there is a skew in the distribution of unique words. Figure 3 illustrates one metric, i.e., the increase in the number of unique words as we increase the number of documents (and words) in the 2 collections. The number of unique words increases steeply for the *Spinn3r* collection whereas the number appears to increase more gradually for **TDT4** and to eventually taper off. The skew is more noticeable in *Spinn3r* since the documents are smaller and the word counts are correspondingly lower. The skewed word distribution for *Spinn3r* makes it difficult for topic modeling approaches that rely on word distribution to cre-

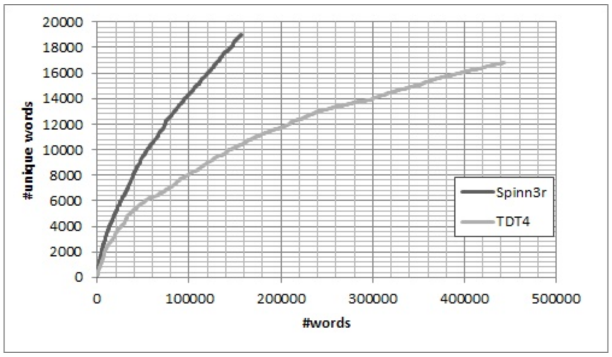

Fig. 3. The number of unique words in the **TDT4** and *Spinn3r* datasets

ate accurate clusters. This same skew benefits **KeyGraph** since an increase in the number of unique keywords correspondingly reduces word co-occurrence. This allows KeyGraph to filter irrelevant documents.

4.2. Comparing the Accuracy of KeyGraph to Other Methods on the TDT4 Dataset

We evaluate two variants as follows:

(1) **KeyGraph**: This a variant of KeyGraph which only uses terms in each document as features.
(2) **KeyGraph+**: This a variant of KeyGraph which uses terms in each document as well as noun phrases and named entities as features.

We compare them with the following best performing topic detection algorithms:

(1) **kNN**: The k-Nearest Neighbor clustering algorithm is a popular clustering solution for TDT tasks[5]. For kNN, we report the results reported by Li et al. (Figure 6 from [13]) under the best threshold.
(2) **GAC**: This is the Augmented Group Average Clustering model proposed by Yang et al. [30]. Similarly for GAC, we report the results reported by Li et al. (Figure 6 from [13]) under the best threshold.
(3) **LDA-VEM**[1]: This LDA variant uses *Variational Expectation Maximization* (with default parameters: α and β will be estimated by the algorithm itself for smoothing over document topic distribution and word distribution, $i = 100$ for the number of expectation maximization iterations, and $k = 71$ for the number of topics which is known for TDT4 dataset.) LDA-VEM also performs hyper-parameter optimization. Similar to LDA-GS, we assign each document to *only* one topic which has the highest probability.
(4) **LDA-GS**[2]: This is the LDA algorithm from the *Mallet* toolkit which uses *Gibbs sampling* (with default parameters: $\alpha = 0.5$ for smoothing over document topic distribution, $\beta = 0.01$ for smoothing over document word distribution, and $k = 71$ for the number of topics which is known for TDT4 dataset.) We note that LDA-GS does hyper-parameter optimization which allows the model to better fit the data. Asuncion et al. [6] showed that for several datasets that contain between 1,000 and 20,000 documents, topic models converge in less than 500 iteration of Gibbs sampling. In our experiments, we selected Mallet's default value $i = 1000$ for the number of Gibbs sampling iterations.
(5) **LDA-GS(p)**[2]: This is a version of **LDA-GS** in which we pre-process the data before feeding it to the Mallet software. Note that Mallet also does some pre-

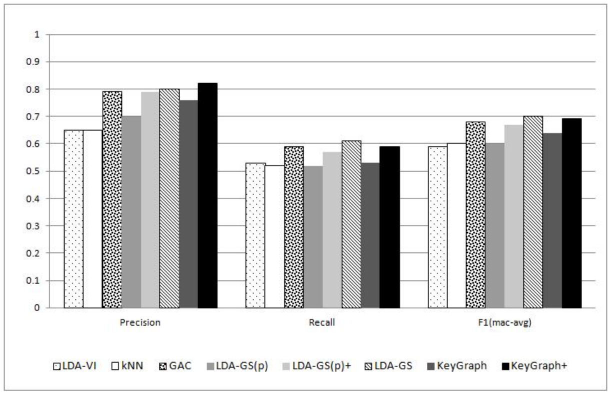

Fig. 4. The performance of the five approaches (kNN, LDA-VEM, LDA-GS, GAC, and KeyGraph) on Dataset 1 (TDT4).

possessing including removing stopwords. In **LDA-GS(p)**, stopwords will be removed first. Next, a Porter stemmer will be applied on the remaining words. Each document will then be represented as a bag of words which will be stored in a text file (the order of the words from the original document may not be preserved). **LDA-GS(p)** uses the same set of features that **KeyGraph** uses.

(6) **LDA-GS(p)+**[2]: This version is a version of **LDA-GS(p)** that uses noun phrases and named entities as features as well. **LDA-GS(p)+** uses the same set of features that **KeyGraph+** uses.

We note that for all variants of LDA and KeyGraph, documents are assigned a probability distribution over topics. However, our gold standard contains only the topic assignments for each document, not the topic distributions. In addition all documents have a topic assigned to them. Hence, we assign each document to at most one topic that has the highest probability. We may filter the documents that their highest probability to any topic is less than 0.1. This is the default threshold in Mallet toolkit.

4.2.1. Evaluation Metrics. Each algorithm may generate any number of clusters, but it is only evaluated on the reference topics. In other words, the cluster that best matches each of the reference topics is used for evaluation as discussed in [13; 30]. We report on the average per-topic *precision* score, the average per-topic *recall* score, and the *macro average* F_1 or F_1(mac-avg) score; The F_1 score is the harmonic mean of precision and recall as follows:

$$F_1 = (2 * precision * recall)/(precision + recall)$$

and the F_1*(mac-avg)* is obtained by computing per-topic *precision* and per-topic *recall* scores first, and then averaging the corresponding F_1 scores [30].

4.2.2. Results. Figure 4 shows the performance of the two variants of KeyGraph and the 6 other algorithms on Dataset 1 (TDT4). The results for all variants of **LDA-GS**, **LDA-VEM**, and **KeyGraph** are obtained by running the algorithms on the TDT4 dataset. The results for **kNN** and **GAC** are from Figure 6 in [13] which was executed on the identical dataset (TDT4).

The results in Figure 4 show that **GAC** and **LDA-GS** outperform **kNN** and **LDA-VEM**, for all three metrics. **KeyGraph+** slightly outperforms **GAC** and **LDA-GS** for precision, scoring `0.82` compared to `0.79` and `0.80` respectively. **LDA-GS** slightly outperforms **GAC** and **KeyGraph+** for recall, scoring `0.61` compared to `0.59` and `0.59`.

Based on the F_1(mac-avg) score, **GAC** with a score of 0.68, **LDA-GS** with 0.70, and **KeyGraph+** with a score of 0.69 have similar performance overall. While the per-topic average precision and recall scores for **GAC**, **LDA-GS**, and **KeyGraph+** appear to outperform **KeyGraph**, **LDA-GS(p)**, and **LDA-GS(p)+**, there was no significant difference as measured by the Wilcoxon Signed Rank test.

Figure 4 shows **LDA-GS** outperforms **LDA-GS(p)** and **LDA-GS(p)**. This suggests that our approach for incorporating named entities and noun phrases into Mallet has a side effect that decrease the performance. However, using noun phrases and named entities as features can slightly improve the performance of both **KeyGraph** and **LDA-GS** algorithms. In summary, the performance of the KeyGraph algorithm on this well known dataset is similar to the gold standard solutions **GAC** and **LDA-GS**.

We note that our comparison did not include the probabilistic model proposed by Li et al. [13] that used the publication date as a feature. Their solution obtained an F_1(mac-avg) score of 0.77 on the **TDT4** dataset. Thus, the probabilistic model solution did have a performance improvement compared to **GAC** and **kNN**. We further note that it significantly dominated **GAC** and **kNN** on another dataset that was collected over a much longer time interval [13]; exploiting the publication date of the articles may explain its superior performance. As we discussed in section 3, we can incorporate publication time by using a dynamic community detection algorithm for topic feature extraction from KeyGraph. We plan to study dynamic network analysis as future research.

We note that the probabilistic model does have the additional drawback that it must estimate the number of events a priori. KeyGraph does not have this limitation. In addition, KeyGraph has many features that make it both scalable and accurate for noisy social media as well be demonstrated next.

4.3. User Evaluation of Social Media Events

We compare the KeyGraph algorithm **KeyGraph+** with a topic modeling solution **LDA-GS** on a social media dataset. We note that **LDA-GS** does not use noun phrases and named entities. However, the experiments in the previous section shows that the improvement in the performance is not statistically significant. In addition, our approach for incorporating named entities and noun phrases into Mallet has a side effect that decrease the performance. Hence, we choose **LDA-GS** for human evaluation since it was the best performing LDA algorithm in the previous section. In this experiments each document will be assigned to at most 1 topic with the highest probability. We filter the documents that their highest probability to any topic is less than 0.1. This is also the default threshold in Mallet toolkit.

4.3.1. Dataset Preparation. We chose the *Spinn3r* Dataset 2 (Table II); with greater than 32,000 articles. We chose the default parameters for LDA-GS; $\alpha = 0.5$ for smoothing over topic distribution, and $i = 1000$ for the number of Gibbs sampling iterations. The number of topics k is unknown a priori for the Spinn3r collection. We chose $k = 200$ for **LDA-GS**. Based on feedback from an expert on topic modeling, a larger number of topics allows LDA to better handle noisy datasets.

KeyGraph identified 94 topics. The average number of relevant documents for the 94 topics identified by KeyGraph was 178, with a standard deviation of 538. In contrast, the average number of relevant documents for the 200 topics identified by LDA-GS was 164, with a standard deviation of 112. These statistics show that the LDA-GS algorithm, even with hyper-parameter optimization, cannot cope with the many irrelevant documents in the dataset. The topics found by LDA-GS are much more uniformly distributed than KeyGraph. KeyGraph could filter out unrelated documents, and the topics are less uniformly distributed.

Since **KeyGraph+** and **LDA-GS** did not produce the same set of topics or the same number of topics, one may argue that comparing their results is not reasonable. To mitigate this criticism, we handpicked an identical number of the *best topics*, i.e., the topics whose keywords and documents were more focused and most specific in their content, for evaluation by humans. These topics were determined by a group of students who did not participate in the evaluation. An alternative was to choose topics of varying sizes. We did not do so since this would appear to benefit **KeyGraph** since most of its topics were small. This has the potential to eliminate good topics of **LDA-GS** that may be large(r).

4.3.2. Evaluation Metrics. Since we do not have the ground truth, we could not directly determine precision and recall [13; 30]. However, we report on *pseudo-precision* and *pseudo-recall*.

To evaluate pseudo-precision, we asked users to vote for the documents assigned to each topic. Both KeyGraph and LDA provide a list of keywords for each topic as part of their models. We used the list of top keywords to describe each topic. We then asked users to vote for each document assigned to each topic; we obtained 3 independent votes for each document. The vote was as follows:

(1) Not Relevant,
(2) Somewhat,
(3) Relevant,
(4) Highly Relevant.

To evaluate pseudo-recall, we asked users to vote for documents that were not assigned to the topics. For each of the topics, the set of *on-topic* documents are those that are selected by some method; the set of *off-topic* documents are the ones that are not selected. We use cosine similarity to select the most similar *off-topic* document to match each *on-topic* document. We use this subset of matching *off-topic* documents for *pseudo-recall*. As users find more of these *off-topic* documents to be not relevant, then *pseudo-recall* increases. We used the list of top keywords identified by KeyGraph and LDA to describe each topic. We asked users to vote for each matching *off-topic* document and obtained 3 independent votes.

4.3.3. Evaluation with Amazon's Mechanical Turk. We used Amazon's Mechanical Turk for the human user evaluation of the performance of the best 14 topics detected by the two algorithms. For each of the 14 topics for each solution, we selected a random subset between 50-150 documents. A total of 972 documents were evaluated for the precision of Top 14 events of KeyGraph and 1132 documents for the precision of top 14 topics of **LDA-GS**.

Tables III and IV present the statistics of the topics/documents and the results of the user evaluation of precision for **LDA-GS** and **KeyGraph+**. As noted in the previous section, KeyGraph can filter irrelevant documents, but LDA-GS cannot. This is reflected in the best 14 topics as well. The 14 topics included 972 documents for **KeyGraph+** and 2307 documents for **LDA-GS**. 12 of the Top 14 topics identified by **KeyGraph+** have less than 100 documents each and 8 have less than 65 documents each. In contrast, the Top 14 topics of **LDA-GS** had a much larger number of documents per topic. Only 4 of the 14 topics has less than 100 documents each and 4 topics had over 200 documents each.

This factor carries through to our second observation on the accuracy of the detected topics. When averaged over all topics, only 40% of the documents for **LDA-GS** received votes of relevant or highly relevant. The inter annotator agreement for **LDA-GS** votes was 74%. In contrast, over 77% of the documents for **KeyGraph+** received votes of

Table III. **LDA-GS**'s pseudo-precision using Mechanical Turk for Spinn3r (32,000 documents)

Event ID	#Docs	#Evaluated Docs	%Not Relevant	%Somewhat	%Relevant	%Highly Relevant
LDAGS1	341	120	33.61	25.56	23.33	17.50
LDAGS2	125	74	45.50	22.52	19.82	12.16
LDAGS3	358	100	27.00	33.00	25.33	14.67
LDAGS4	164	80	39.17	28.33	18.75	13.75
LDAGS5	87	87	44.44	28.74	17.62	9.20
LDAGS6	227	120	30.56	31.39	20.56	17.50
LDAGS7	91	91	55.31	23.81	12.09	8.79
LDAGS8	108	52	46.79	21.79	16.03	15.38
LDAGS9	93	66	36.87	26.77	18.18	18.18
LDAGS10	158	58	31.03	36.78	19.54	12.64
LDAGS11	139	79	44.73	25.32	19.41	10.55
LDAGS12	197	100	37.33	24.67	24.00	14.00
LDAGS13	99	60	48.89	19.44	13.33	18.33
LDAGS14	120	45	43.70	28.15	17.04	11.11
Total	**2307**	**1132**	**49.91**	**33.24**	**23.41**	**17.12**
	Inter Annotator Agreement = 74%					

Table IV. **KeyGraph+**'s pseudo-precision using Mechanical Turk for Spinn3r (32,000 documents)

Event ID	#Docs	#Evaluated Docs	%Not Relevant	%Somewhat	%Relevant	%Highly Relevant
KEYGRAPH1	83	83	2.81	24.10	53.01	20.08
KEYGRAPH2	87	87	2.30	17.62	62.84	17.24
KEYGRAPH3	141	141	0.71	19.15	57.45	22.70
KEYGRAPH4	46	46	4.35	23.91	52.90	18.84
KEYGRAPH5	63	63	1.06	18.52	57.67	22.75
KEYGRAPH6	39	39	1.71	14.53	64.10	19.66
KEYGRAPH7	81	81	3.70	18.11	55.97	22.22
KEYGRAPH8	41	41	4.88	18.70	55.28	21.14
KEYGRAPH9	51	51	3.92	22.88	50.98	22.22
KEYGRAPH10	62	62	4.30	19.35	52.15	24.19
KEYGRAPH11	31	31	3.23	17.20	59.14	20.43
KEYGRAPH12	39	39	1.71	18.80	54.70	24.79
KEYGRAPH13	77	77	3.46	20.78	54.98	20.78
KEYGRAPH14	131	131	3.82	20.10	51.91	24.17
Total	**972**	**972**	**3.00**	**19.55**	**55.93**	**21.52**
	Inter Annotator Agreement = 81%					

relevant or highly relevant. The inter annotator agreement for **KeyGraph+** votes was 81%.

For *pseudo-recall*, users voted for the Top 60 matching *off-topic* documents for each topic. A total of 840 documents were evaluated for each method. Tables V and VI

Table V. **LDA-GS**'s pseudo-recall using Mechanical Turk for Spinn3r (32,000 documents)

Event ID	**#Evaluated Docs**	**%Not Relevant**	**%Somewhat**	**%Relevant**	**%Highly Relevant**
LDAGS1	60	70.00	16.11	10.56	3.33
LDAGS2	60	66.67	15.00	11.11	7.22
LDAGS3	60	73.89	10.00	9.44	6.67
LDAGS4	60	90.00	3.89	3.89	2.22
LDAGS5	60	63.89	23.33	6.67	6.11
LDAGS6	60	90.00	3.89	3.89	2.22
LDAGS7	60	90.56	1.67	3.89	3.89
LDAGS8	60	79.44	8.89	8.89	2.78
LDAGS9	60	74.44	18.33	5.56	1.67
LDAGS10	60	61.67	22.22	10.56	5.56
LDAGS11	60	56.67	23.89	14.44	5.00
LDAGS12	60	92.22	5.00	1.11	1.67
LDAGS13	60	33.89	22.22	32.78	11.11
LDAGS14	60	85.56	8.33	5.56	0.56
Total	**840**	**73.49**	**13.6**	**9.17**	**4.29**
	Inter Annotator Agreement = 86%				

present the statistics of the topics/documents and the results. Note that as the number of *off-topic* documents judged to be relevant decreases, *pseudo-recall* increases; this is because these were *off-topic* documents that were not chosen by some method. When averaged over all topics, about 13% of documents for **LDA-GS** received votes of relevant or highly relevant. The inter annotator agreement for **LDA-GS** votes was 86%. Similarly, about 11% of documents for **KeyGraph+** received votes of relevant or highly relevant. The inter annotator agreement for **KeyGraph+** was 89%.

Thus, since we do not have *ground truth* for this dataset, we compared pseudo-precision and pseudo-recall. These results show that **KeyGraph+** can filter noise and that the precision of **KeyGraph+** appears to significantly outperform **LDA-GS** for this noisy data collection. The improved precision of **KeyGraph+** does not come at the expense of lower recall; the *pseudo-recall* for **KeyGraph+** is slightly better than for **LDA-GS**.

4.4. Running Time of Methods

We discuss the complexity of KeyGraph and the LDA topic modeling with Gibbs sampling (**LDA-GS**). We then compare the execution time of KeyGraph and **LDA-GS** on a large data collection.

Let N be the number of documents in the collection, M the number of unique words in the collection and $M << N$. K represents the number of topics. If D shows the maximum number of unique keywords in a document, the complexity of the three steps of the KeyGraph algorithm is as follows: Building the KeyGraph is done in $O(N*D+M^2)$ operations, which is the running time of a single pass through all documents in the collection and building the KeyGraph (The number of nodes in a KeyGraph is limited to M and the number of edges to M^2). The running time of the second step is equal to the running time of the community detection algorithm. We used betweenness centrality approach which is running in $O(M^3)$ operations[17]. The final step is another pass

Table VI. **KeyGraph+**'s pseudo-recall using Mechanical Turk for Spinn3r (32,000 documents)

Event ID	**#Evaluated Docs**	**%Not Relevant**	**%Somewhat**	**%Relevant**	**%Highly Relevant**
KEYGRAPH1	60	77.78	11.67	6.67	3.89
KEYGRAPH2	60	85.56	9.44	2.78	2.22
KEYGRAPH3	60	40.56	28.89	23.89	6.67
KEYGRAPH4	60	85.00	7.22	5.00	2.78
KEYGRAPH5	60	73.89	15.00	8.89	2.22
KEYGRAPH6	60	48.89	19.44	21.11	10.56
KEYGRAPH7	60	92.78	1.67	1.67	3.89
KEYGRAPH8	60	84.44	8.33	5.56	1.67
KEYGRAPH9	60	86.11	9.44	3.33	1.11
KEYGRAPH10	60	71.11	17.22	7.78	3.89
KEYGRAPH11	60	88.89	7.78	3.33	0.00
KEYGRAPH12	60	85.00	8.89	4.44	1.67
KEYGRAPH13	60	82.78	8.89	5.00	3.33
KEYGRAPH14	60	87.78	7.78	2.78	1.67
Total	**840**	**77.90**	**11.55**	**7.30**	**3.25**
	Inter Annotator Agreement = 89%				

through the data collection which will be done in $O(N * D * K)$ operations. Hence, the running time can be summarized as the following:

- Building the KeyGraph: $O(N * D + M^2)$
- Topic feature extraction: KeyGraph community structure analysis using betweenness centrality metric [17]: $O(M^3)$
- Topic document assignment: $O(N * D * K)$
- **Total Running Time:** $O(N * D * K + M^3)$

The total running time is linear in terms of the number of documents. Notice that second term in the complexity of KeyGraph, M^3, is the running time of the community detection on the KeyGraph which is done by betweenness centrality approach. There are faster community detection algorithm such as the algorithm of Radicchi et al. [20] and the *Kernighan-Lin* algorithm [11] which run in $O(M^2)$ operations [17]. Using the faster community detection algorithm can reduce the complexity of the algorithm specially for small datasets which M is not much smaller than N.

Next we discuss the complexity of the topic modeling approaches. Let I be the number of iterations for LDA with Variational Expectation Maximization. For each document, in each iteration, the complexity of the computation is $O(D * K)$. Usually the inference iterations stops when ELBO (variational likelihood lower bound) is met. Hence, the complexity of LDA with Variational Expectation Maximization is $O(N * D * K * I)$. Similarly for LDA with Gibbs sampling, let I be the number of iterations, and L be the maximum number of keywords in a document or the length of a document. For each document, in each iteration, the complexity of the computation is $O(L * K)$. Gibbs sampling typically needs to be done for an absurdly large number of iterations (about hundreds or thousands iteration). Hence, the complexity of LDA with Gibbs sampling is $O(N * L * K * I)$. In summery, the complexity of a LDA Topic modeling algorithm is as follows:

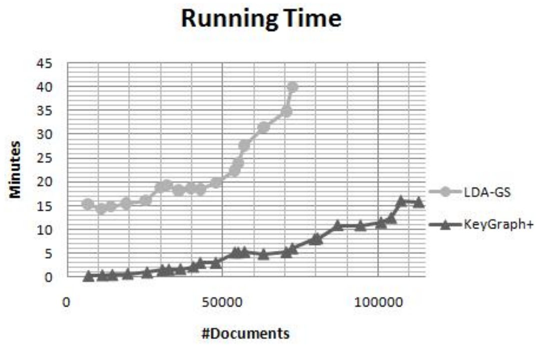

Fig. 5. The running time of KeyGraph Algorithm versus LDA-GS implemented in Mallet toolkit

— LDA with Variational EM: $O(N * D * K * I)$
— LDA with Gibbs Sampling: $O(N * L * K * I)$

where I is the number of iterations.

In practice the number of iterations depends on many factors such as the number of documents and topics as well as the length of the documents. Because this process is all stochastic, it is hard to give rigorous bounds for the number of iterations before convergence. Hence, we compare the running time of the KeyGraph and LDA-GS in our experiments. Figure 5 reports on the execution time of **KeyGraph+** and **LDA-GS** as we increase the number of documents; we use Spinn3r Dataset. We create subsets that are an increasing time slice of this dataset where the first subset is 1 day of data, the second is 2 days of data, etc., up to 25 days of data. The number of documents varied from 6,559 to 113,000 and is shown on the X axis of Figure 5. The experiment was performed on a machine with an Intel Core 2 Due 2.93GHz processer and a 4GB memory running Linux.

Figure 5 shows the total running time of **LDA-GS** and **KeyGraph+**. We stopped running **LDA-GS** after 72,000 documents since it was very slow. We ran **KeyGraph+** on up to 113,000 documents. The execution time of **KeyGraph+** appears to increase in a linear manner as predicted by the complexity analysis. In comparison, the execution time of **LDA-GS** increases in a non-linear manner. As the number of documents exceeds 72,000 the running time of **LDA-GS** is almost an order of magnitude greater than **KeyGraph+**.

4.5. Model Robustness

We studied the sensitivity of KeyGraph algorithm with respect to the parameters used in the algorithm. If the result of an algorithm is very sensitive to the value of a parameter, then it will not be easily applicable to other domains and dataset. For algorithms that have multi steps, if a sensitive step is not done perfectly it may affect the result of the following steps significantly. Hence, we are interested to analyze the robustness of our proposed model to its parameters. Recall KeyGraph community structure analysis and extracting topic features from KeyGraph is the core of our model and robustness of our algorithm to these steps is very important. We report the robustness of the algorithm on three parameters: The first two parameters are α and β which control the termination of the community detection algorithm. As our community detection algorithm removes edges with the high betweenness centrality, α and β identify where to stop splitting the graph. These are the termination parameters which are required for any network community detection algorithms. For the betweenness community detec-

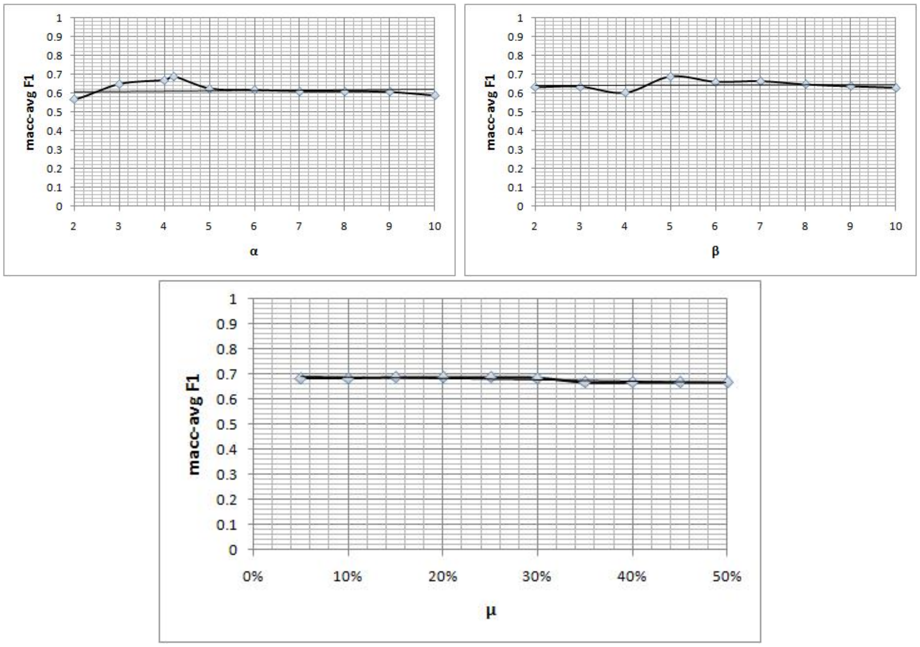

Fig. 6. Sensitivity analysis for parameter α and β that identify when to stop splitting the communities for community detection on keyGraph, as well as μ which indicates the minimum overlap of cluster to be merged in the last step of our algorithm.

tion algorithm that we used in our experiments, we stop removing edges if the highest betweenness centrality score is less than a threshold θ (which uses α) or the size of the community is less than β. θ is defined as the logarithmic function of the size of a KeyGraph G: $\theta = \alpha \log |G| + 1$.

The third parameter, μ, controls the process of merging topics with an overlap in their document set in the last step of KeyGraph algorithm. If the document set overlap of two topics is greater than μ, the topics will be merged.

Intuitively, KeyGraph model should be very robust to the value of α and β since if a community is mistakenly broken into two or communities, it will be resolved later in the last step which overlapping topics will be merged.

Here we did an experiment on the TDT4 dataset to evaluate the robustness of KeyGraph. We ran the KeyGraph algorithm on a very large range of the values for all three parameters and report F_1(mac-avg) for each run. Figure 6 shows the robustness of our model on the TDT4 dataset with respect to parameter α and similarly β. We can see that for a large range of α and β values the F_1(mac-avg) on the TDT4 dataset decrease only a little bit. These results show that our KeyGraph algorithm is very robust to the result of the community detection. Figure 6 also shows that our model is completely robust to the value of μ which indicated the minimum overlap of cluster to be merged in the last step of our algorithm.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we present a graph analytical approach for fast topic detection, KeyGraph. We use communities in a word co-occurrence graph as a proxy for topics. Using an extensive evaluation we demonstrate the accuracy and scalable performance of the KeyGraph algorithm, in particular for noisy social media datasets. Given the success of this heuristic solution, KeyGraph, we plan to fully explore an extension to topic modeling to consider both word and keyword co-occurrence distributions.

KeyGraph is naturally time dependent. Our current approach is based on static community detection and we use an aggregated KeyGraph. The timestamp is therefore very important information for event detection which is not exploited. We note that even without the important time dependent information, the accuracy of KeyGraph is similar to the best performing topic detection models on clean data, where no model uses the timestamp. Further, KeyGraph outperforms the gold standard models for noisy data.

Li et al. [13] showed that their method can exploit time dependent information and can dominate on a data collection that is obtained over a longer interval. This result motivates us to retain timestamps in the KeyGraph, and to explore methods from dynamic network analysis. Thus, in future work, we will explore the benefits of dynamic community detection algorithms for event detection in a KeyGraph. Since these methods are computationally expensive, we will study the tradeoff of added accuracy and determine incremental approaches to update the KeyGraph.

REFERENCES

C implementation of variational expectation maximization for latent dirichlet allocation (lda). *Http://www.cs.princeton.edu/ blei/lda-c/index.html*.

Machine learning for language toolkit. *Http://mallet.cs.umass.edu/topics.php*.

Topic detection and tracking(tdt) project. *Http://www.nist.gov/speech/tests/tdt/*.

The future of social networking: Understanding market stratigic and technology developments. *In Datamonitor*, 2007.

J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR98: Proceedings of ACM SIGIR*, 1998.

A. Asuncion, M. Welling, P. Smyth, and Y.-W. Teh. On smoothing and inference for topic models. In *UAI*, 2009.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

T. Brants, F. Chen, and A. Farahat. A system for new event detection. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 330–337, Toronto, Canada, 2003.

D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. pages 318–329, 1992.

Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214, 2007.

B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304, United Kingdom, 2004.

Z. Li, B. Wang, M. Li, and W.-Y. Ma. A probabilistic model for retrospective news event detection. In *SIGIR05: Proceedings of ACM SIGIR*, Brazil, 2005.

M. Mori, T. Miura, and I. Shioya. Extracting events from web pages. In *AISTA'04: Proceedings of the International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA)*, 2004.

M. Mori, T. Miura, and I. Shioya. Topic detection and tracking for news web pages. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 338–342, 2006.

D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 100–108, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):321–330, March 2004.

Y. Ohsawa, N. E. Benson, and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *ADL '98: Proceedings of the Advances in Digital Libraries Conference*, page 12, Washington, DC, USA, 1998. IEEE Computer Society.

R. Prabowo, M. Thelwall, I. Hellsten, and A. Scharnhorst. Evolving debates in online communication: a graph analytical approach. *Internet Research: Electronic Networking Applications and Policy*, 18(5):520–540.

F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, March 2004.

N. Ruan, R. Jin, V. Lee, and K. Huang. Dynamic module discovery in temporal complex networks. In *2nd Intl. Workshop on Analysis of Dynamic Networks, in conjunction with SIAM Intl. Conf. on Data Mining*, 2009.

N. Ruan, R. Jin, V. Lee, and K. Huang. A sparsification approach for temporal graphical model decomposition. In *ICDM'09: International Conference on Data Mining*, 2009.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, volume 24, pages 513–523, 1988.

H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Proceedings of International Conference on Weblogs and Social Media (ICWSM)*, 2009.

H. Sayyadi, S. Salehi, and H. Abolhassani. Nesrec: News meta-search result clustering". In *n Proceeding of CIS2E 06 The International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering*, pages 173–178, USA, 2006.

M. Steyvers and T. Griffiths. *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007.

J. Tantrum, A. Murua, and W. Stuetzle. Hierarchical model-based clustering of large datasets through fractionation and refractionation. *Information Systems*, 29(4):315 – 326, 2004. Knowledge Discovery and Data Mining (KDD 2002).

H. Toda and R. Kataoka. A search result clustering method using informatively named entities. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 81–86, Bremen, Germany, 2005.

Y. Yang, N. Bansal, W. Dakka, P. G. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *WSDM*, pages 34–43, 2009.

Y. Yang, T. Pierce, and J. G. Carbonell. A study on retrospective and on-line event detection. In *SIGIR98: Proceedings of ACM SIGIR*, 1998.