HG-RANK: A Hypergraph-based Keyphrase Extraction for Short Documents in Dynamic Genre

Abdelghani Bellaachia
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
bell@gwu.edu

Mohammed Al-Dhelaan*
Department of Computer Science
The George Washington University
Washington, DC 20052, USA
mdhelaan@gwu.edu

ABSTRACT

Conventional keyphrase extraction algorithms are applied to a fixed corpus of lengthy documents where keyphrases distinguish documents from each other. However, with the emergence of social networks and microblogs, the nature of such documents has changed. Documents are now of short length and evolve topics which require specific algorithms to capture all features. In this paper, we propose a hypergraph-based ranking algorithm that models all the features in a random walk approach. Our random walk approach uses weights of both hyperedges and vertices to model short documents' temporal and social features, as well as discriminative weights for word features respectively, while measuring the centrality of words in the hypergraph. We empirically test the effectiveness of our approach in two different data sets of short documents and show that our approach has an improvement of 14% to 25% in precision over the closest baseline in a Twitter data set and 10% to 27% in the Opinosis data set.

Categories and Subject Descriptors

I.2.7 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing—*Text analysis*

Keywords

Text hypergraphs; Keyphrase extraction; Random walks; Short documents; Hypergraph random walks

1. INTRODUCTION

Short text messages are ubiquitous nowadays in social networks and across the web. Regardless of the length limitation, the size restriction did not limit the popularity of social interaction through such messages. Twitter, for instance, has more than 200 million active users each month[1]. Such high popularity necessitates ranking systems capable of measuring the importance of keywords and keyphrases within such limited length to facilitate search, indexing, and detecting trends. By finding salient terms, tasks such as summarization and text visualization become feasible. However, the dynamic nature of social microblogs makes ranking a non-trivial task.

Microblogs have a dynamically changing content that needs specially designed algorithms for keyphrase extraction. Descriptive keyphrases are keyphrases that signify topics in a document and help differentiate it from other documents in the corpus. However, the social aspects and evolution of topics in a social media genre make it rather difficult to find keyphrases. Most keyphrase extraction algorithms do not account for the temporal and social attributes when finding keyphrases since they are designed for static documents corpora. Therefore, a number of interesting research questions arise in social microblogs where topics change frequently. If the content is dynamically changing, then can we rely on conventional keyphrases extraction approaches? How can we account for the temporal and social attributes in social media for keyphrase extraction?

In this article, we present a hypergraph-based algorithm, called HG-RANK, that is capable of modeling temporal and social aspects in addition to discriminative weights. A hypergraph is a generalization of graphs were edges have a set of vertices (called hyperedges) instead of two nodes. We define a lexical hypergraph where vertices are distinct words and hyperedges are short documents that contain the words. We model the temporal and social attributes of documents as hyperedge weights to reflect the attributes over the document's keywords, and we model discriminative term weights as vertex weights to give the model the ability of recognizing topical terms. We design a weighted random walk over the hypergraph to measure the centrality of keywords taken into account all the aforementioned features.

To rank vertices in a hypergraph, we generalize a probabilistic random walk suitable for a weighted hypergraph structure. The surfer considers the weights of both vertices and hyperedges for transitioning. The intuition is that the surfer will prefer words that has the following properties. The words belong to a recent document, and they exist in a document that has attracted social users, for in-

*This author is sponsored by King Saud University, Saudi Arabia

Published as part of the #Microposts2014 Workshop proceedings, available online as CEUR Vol-1141 (http://ceur-ws.org/Vol-1141)

#Microposts2014, April 7th, 2014, Seoul, Korea.

[1]https://twitter.com/twitter/status/281051652235087872

stance re-tweet in Twitter. Additionally, the surfer will prefer topically discriminative words capable of finding accurate keyphrases.

The contribution of this paper can be summarized as follows:

- We propose a new hypergraph approach to jointly model temporal and social features within the hypergraph structure. This model is capable of recognizing the importance of time and social features that are important in a dynamic genre.
- The hypergraph-based HG-RANK algorithm is the first graph-based approach for keyphrase extraction that considers the high-order relation between words instead of a pair-wise relation as in conventional graph-based keyphrase extraction.
- We evaluate our approach with two different data sets Twitter and Opinosis. We show the effect of each dimension on the task of keyphrase extraction.

The rest of the paper is organized as follows. A discussion of the related work is in Section 2. We define the hypergraph notation needed for explaining the proposed approach in Section 3. The proposed approach will be thoroughly explained in Section 4. Section 5 will describe the data and experimental results. The paper conclusion is in Section 6.

2. RELATED WORK

Our work is related to three different research areas, namely: temporal and social aspects for keyphrase extraction, graph-based keyphrase extraction, and hypergraphs. This work bridges such areas for the task of keyphrase extraction in a unified framework.

The emergence of social networks has motivated researchers to examine the inclusion of temporal and social dimensions into search [27][16][13], summarization[24][23], and keyphrase extraction[28][11]. Yu et al. proposed to combine the temporal dimension into a PageRank[6] approach for ranking research publications considering their publication time[27]. Wan proposed a time-aware summarization algorithm over a lexical graph[24]. A probabilistic approach for personalized temporally-aware tweets summarization is proposed in [22] For including social aspects, Zhao et al. proposed to do keyphrase extraction while they used an interestingness score for capturing social attribute[28]. A multi-document summarizer that takes into account social features is proposed in [17]. Moreover, a lexical graph expansion for extracting keyphrases through social hashtags is shown in [2].

A number of graph-based keyphrase extraction approaches have been proposed. TextRank[20], LexRank[9], NE-Rank[3], SingleRank[25], and TopicRank[5]. These algorithms leverage a random walk to calculate the centrality of either words or sentences. For instance, NE-Rank considers node weights being tf-idf of words and edge weight being freqancy of co-occurrence of pairs of words. However, they all use simple graphs not hypergraphs. In this paper, we consider a high-order co-occurrence relation modeled in a hypergraph.

Hypergraph random walks have been proposed in [29][1]. We further extend the aforementioned approaches in this work to include vertex weights. Wang et al. proposed to use a semi-supervised ranking approach based on Zhou et al.[29] for ranking sentences which they used for text summarization[26]. Li et al.[14][15] proposed a semi-supervised keyphrase ranking over hypergraphs based on Zhou et al.[29] defintions. They proposed using semantic connection between phrases(vertices) to form hyperedges using external knowledge sources as in Wikipedia. Our work is different in the following matter: we use a completely unsupervised approach for ranking keywords instead of sentences or phrases which may not be easy to find in social snippets, we propose a new weighted random walk that uses both hyperedges and vertices weights, and we include temporal and social attributes in the ranking. Finally, unlike exciting approaches for semi-supervised hypergraph ranking our ranking approach is query independent and thus unsupervised.

3. NOTATIONS AND DEFINITIONS

Let $HG(V, \mathcal{E})$ be a hypergraph with the vertex set V and the set of hyperedges $\mathcal{E}$. A hyperedge e is a subset of V where $\cup_{e \in \mathcal{E}} e = V$. Let $HG(V, \mathcal{E}, w)$ be a weighted hypergraph where $w : \mathcal{E} \rightarrow \mathbb{R}^+$ is the hyperedge weight. A hyperedge e is said to be incident with v when $v \in e$. A hypergraph has an incidence matrix $H \in \mathbb{R}^{|V| \times |E|}$ as follows:

$$h(v, e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{if } v \notin e \end{cases} \quad (1)$$

The vertex and hyperedge degree are defined as follows:

$$d(v) = \sum_{e \in E} w(e) h(v, e) \quad (2)$$

$$\delta(e) = \sum_{v \in V} h(v, e) = |e| \quad (3)$$

D_e and D_v are the diagonal matrices representing the degrees of hyperedges and vertices, respectively. W_e is the diagonal matrix with the hyperedge weights.

4. PROPOSED APPROACH

The HG-RANK model captures keyphrases using a hypergraph structure where it is possible to inherently model social and temporal features. These features are embedded as a hyperedge weight that represents a specific short document. In essence, we model each short text document d_i as a bag-of-words model with distinct keywords $d_i = \{k_1, k_2, ..., k_s\}$. A collection of documents $D_i = \{d_1, d_2, ..., d_n\}$ is then represented as a lexical hypergraph in the following manner. We represent each short document as a hyperedge, and each keyword as a distinct vertex.

In this section, we will describe the HG-RANK algorithm in more depth. First, the calculation and insertion of the temporal and social attributes is going to be explained. Second, the vertex weights will be explained. Third, we will define the random walk ranking approach on the lexical hypergraph to rank keywords. Finally, we will discuss our approach on extracting keyphrases.

4.1 Modeling Temporal & Social Features

Temporal attributes in a dynamic genre as a microblogging social network or news trends is an important dimension to understand evolving topics and keyphrases. We measure the temporal effect as a ranking function for short documents. The more recent the document, the higher the temporal rank will be. Similar to [27][24], we measured the

temporal effect as the following:

$$R_{time}(d_i) = Q^{(c-y_i)/24} \quad (4)$$

Where c and y_i denote the current time and the document d_i publication time, respectively. $(c-y_i)$ is the time interval between current and publication time in hours. We divide by 24 to show the difference of publication time and current time in number of days. Q is a decay rate parameter with values $0 < Q < 1$. Moreover, the Q value is inversely proportional to favoring recent documents. When Q is closer to 0, the ranking favors very recent documents over old ones. On the other hand, when Q is closer to 1, the ranking is less focused on new documents. In our experiments, we set Q to 0.5.

For the social effect, we measure the social dimension of documents as a ranking function. The more popular or shared the document, the higher the social rank will be. For example in Twitter, tweets that are re-tweeted frequently should be more important than a tweet without re-tweets. This is similar to other social networks with the "like" feature as in Facebook or product reviews. We calculate the social ranking as follows:

$$R_{social}(d_i) = \frac{s_i + 1}{\sum_e s_e + 1} \quad (5)$$

Where s_i is the counter of social feature (counts of re-tweet or likes) for document d_i. $\sum_e s_e$ is the sum of all social features across all documents (total number of re-tweets for example). Moreover, we added one smoothing to avoid canceling out documents with no social attributes.

Now we tie both temporal and social features together in one ranking function as follows:

$$w(d_i) = \lambda R_{social}(d_i) + (1-\lambda) R_{time}(d_i) \quad (6)$$

λ is a smoothing parameter with $0 < \lambda < 1$ to trade off the effect of temporal aspects and social aspects. We experimented with different values for λ which will be discussed in the experiment section. The final documents rank $w(d_i)$ will be embedded in the hypergraph as a *hyperedge weight* to reflect documents' importance over keywords. The intuition behind embedding temporal and social features in the ranking scheme is that they are essential for capturing keyphrases in a dynamic genre. In a dynamic genre, as in Twitter, the content rapidly changes with time. Hence, the keyphrases tend to change as well. Conventional keyphrase extraction algorithms do not consider the time dimension in finding keyphrases which make them insufficient for the task. Moreover, the social aspect is important to capture keyphrases of trendy topics that social network users find interesting. An interesting topic in social media will more likely be searched compared to other topics which makes it important to find its keyphrases. We will discuss vertex weights in the next section.

4.2 Modeling Discriminative Weights as Vertex Weights

Graph-based approaches base the ranking on the relational structure of co-occurring words. Such ranking is great on capturing the semantic relation between words. However, there is no evidence that graph-based ranking approaches are able to capture discriminative words. To enhance the hypergraph-based ranking algorithm, we use a discriminative weighting scheme tf-idf as vertex weights before we start the random walk. This injection of tf-idf weights will add a discriminative perspective for calculating the rank though a random walk approach. However, when applied to short text documents, tf-idf fails to capture descriptive terms due to sparsity of features (short length). To circumvent the sparsity problem, we aggregate short documents to a virtual larger ones and then calculate tf-idf scores. A larger virtual document δ is the concatenation of smaller documents d which is $\delta_t = \{d_1 + d_2 + ... + d_n\}$. In Section 5.3, different approaches for aggregation are described in more depth. We measure the normalized tf-idf over the larger documents being the set of $D = \{\delta_1, \delta_2, ..., \delta_n\}$. The tf-idf is measured as follows:

$$w(v_i)_{tf\text{-}idf} = \frac{tf(v_i)}{N_w} \cdot log \frac{N}{df(v_i)} \quad (7)$$

Where $tf(v_i)$ as the term frequency on the document δ and N_w is sum of all words occurrences in document δ for normalization. N is the number of documents in the larger document set D, and $df(v_i)$ is the number of larger documents in D that contain the term v_i. We will discuss the hypergraph ranking algorithm HG-Rank in detail in the next section.

4.3 HG-Rank: Ranking in a Hypergraph

To rank vertices in a hypergraph, we generalize a random walk process for hypergraphs. A random walk process is the transitioning between vertices in a graph by starting at a given vertex and moving to another neighboring vertex after each discrete time step t. We can imagine vertices as a set of states $\{s_1, s_2, ..., s_n\}$ and the transitioning to be a finite Markov chain $\mathcal{M}$ over these states. The transition probability calculated as $P(u,v) = Prob(s_{t+1} = v | s_t = u)$ which means that the chain $\mathcal{M}$ will be at v at time $t+1$ given that it was observed at u at time t. The Markov chain herein is *homogeneous* which means that the transition probability is independent of time t. Note that for any vertex u we have $\sum_v P(u,v) = 1$. Since $\mathcal{M}$ is homogeneous with probabilities computed over only a single transition, we can then define a transition matrix $P \in \mathbb{R}^{|V| \times |V|}$ for all moves. The transition matrix P captures the transition between vertices which shows the behavior of a surfer randomly moving between vertices according to such probabilities. Next we will show how we define the random walk in hypergraphs.

In simple graphs[2], the random walk process is clear by simply choosing an edge with a probability to a destination vertex. However, it is not the case in hypergraphs where the structure of the graph is substantially different demanding a more general walk. For instance, in a hypergraph, a hyperedge could have more than two end-point vertices $\delta(e) \geq 2$. To generalize the random walk process in hypergraphs, we model the walk as the transition between two vertices that are incident to each other in a *hyperedge* instead of a normal edge. In essence, the random walk is seen to be a two-step process, instead of one, which is the following: the random surfer first chooses a hyperedge e incident with the current vertex u. Then the surfer picks a destination vertex v within the chosen hyperedge satisfying the following $u, v \in e$. The

[2]By simple graphs, we mean graphs (not hypergraphs) with edges that are unique pair of vertices. Not to be confused with simple vs. multigraph

random walk in hypergraph is said to be more general since the random walk in a normal graph is a special case where there is only a single destination vertex v associated with a given normal edge incident with u where in a hypergraph we can have more vertices to choose from. The hypergraph random walk process can be defined as a Markov chain where the vertex set is the state set of the chain similar to a normal graph. At each time step t the surfer moves in the incident hyperedge to another vertex.

In this paper, we try to seek a general definition of a random walk in a weighted hypergraph where not only hyperedges have weights, but vertices as well. In such a case, the random walk process is extended to leverage both hyperedges' and vertices' weights. We define the vertex's weight across all incident hyperedges to be a feature vector

$$\vec{v}_w = \{w(v_{e1}), w(v_{e2}), ..., w(v_{d(v)})\} \tag{8}$$

Where we have a different vertex's weight for every hyperedge e that contain vertex v. We describe the proposed random walk process as the following. Starting from a vertex u, the surfer chooses a hyperedge e incident with u proportional to the hyperedge weight $w(e)$. Then, the surfer, also chooses a vertex v proportional to the vertex weight within the hyperedge where we consider the weight in the current hyperedge only. Let us define a weighted hypergraph incident matrix $H_w \in \mathbb{R}^{|V|\times|E|}$ where we have the following:

$$h_w(v,e) = \begin{cases} w(v_e) & \text{if } v \in e \\ 0 & \text{if } v \notin e \end{cases} \tag{9}$$

Therefore, we redefine the hyperedge degree to be as follows:

$$\delta(e_w) = \sum_{v\in V} h_w(v,e) \tag{10}$$

We can now calculate the transition matrix P as follows:

$$P(u,v) = \sum_{e\in E} w(e) \frac{h(u,e)}{\sum_{\hat{e}\in\mathcal{E}(u)} w(\hat{e})} \frac{h_w(v,e)}{\sum_{\hat{v}\in e} h_w(\hat{v},e)} \tag{11}$$

Or in matrix notation:

$$P = D_v^{-1} H W_e D_{ve}^{-1} H_w^T$$

Where $h_w(v,e)$ is the weight of the destination vertex v in hyperedge e. D_v is the diagonal matrix of the *weighted* degree of vertices as in formula 2. W_e is the diagonal matrix of the hyperedge weights. D_{ve} is the diagonal matrix for weighted degree of hyperedges as in formula 10. Note that the transition matrix P is *stochastic* where we have every row sums to 1.

After calculating the transition matrix P, we now explain the stationary distribution π of a random walk. The stationary distribution can be calculated by starting with initial column vector $\vec{v}_0 \in \mathbb{R}^{|V|\times 1}$ with equal probabilities $1/|V|$ summing to 1. We first multiply the transition matrix P^T(where P^T is a column stochastic matrix for clarity) by the initial column vector $\vec{v}_0$ yielding $\vec{v}_1 = P^T\vec{v}_0$. Then, we iterate until the vector $\vec{v}$ stops changing. The reason of multiplying the probability distribution vector $\vec{v}$ by the transition matrix P^T gives us the next step distribution $\vec{x} = P^T\vec{v}$ can be explained as follows. Let x_i be the probability of being at the current vertex i. Then we have the following: $x_i = \sum_j p_{ij}v_j$ where v_j being the probability of the surfer being at node j previously, and p_{ij} is the probability of moving from j to i.

The probability distribution vector $\vec{v}$ stops changing after n steps if the random walk is *ergodic*. A random walk is ergodic when the following conditions are met: 1) the chain is *irreducible*, for any two states $s_i, s_j \in \mathcal{M}$ they must satisfy $P(s_i, s_j) > 0$. Also, 2) the chain is *aperiodic*, where the greatest common divisor of every state $\{t : P_t(s_i, s_i) > 0\}$ is 1. To guarantee irreducibility and aperiodicity, we use the PageRank algorithm [6]. The algorithm uses the idea of *teleporting* which will restart the random walk process making it useful for the previous conditions. The teleporting is depicted with a small probability called the *damping factor* α. It also makes sure to make the graph irreducible since the random walker always has the probability of teleporting to any other node.

$$\vec{v}_{(i+1)} = \alpha P^T\vec{v}_{(i)} + (1-\alpha)\vec{e}/n \tag{12}$$

The damping factor α is set to 0.85. n is the number of nodes in the graph. $\vec{e} \in \mathbb{R}^{n\times 1}$ is a vector of all elements being 1. $\alpha P^T\vec{v}$ means that the random walker will choose to go with one of the incident hyperedges. $(1-\alpha)\vec{e}/n$ represents a vector of an introductory probabilities with each entry being $(1-\alpha)/n$ to teleport the random walk to a new node.

4.4 Extracting Keyphrases

We tag keywords with their Part of Speech (POS) tags. Then, we extract keyphrases that are noun phrases since it has been shown that most keyphrases annotated by human happen to be noun phrases[12][18][25]. We look for patterns as adj+nouns or all nouns and filter out the rest. Then, we have a candidate list of keyphrases based on the syntactic filtering that need to be ranked. A keyphrase ph is modeled as a collection of keywords k as $ph = \{k_1, k_2, ..., k_n\}$. To rank a keyphrase, most approaches aggregate the ranks of the keywords as follows:

$$R(ph) = \sum_{k_i\in ph} R(k_i) \tag{13}$$

However, such approach will be biased towards longer phrases. To overcome such bias, we normalized based on the length of the keyphrase as follows:

$$R(ph) = \frac{\sum_{k_i\in ph} R(k_i)}{n} \tag{14}$$

Where n is the keyphrase length. Moreover, we removed phrases that cross over syntactic boundaries as they cannot be a comprehensible keyphrase. We also removed any keyphrase that appears less than f times. We experimented with different values of f and found out that $f = 5$ shows the best keyphrases in our data. Next we will describe the experimental design in depth and all comparisons.

5. EXPERIMENT

This section explains the experimental setup for the hypergraph ranking framework HG-Rank. The effectiveness of our approach is demonstrated by conducting several experiments comparing our method to different baselines. First, the data sets used in this experiment are explained thoroughly. Second, the necessary preprocessing steps are illustrated in detail. Third, the experimental setup is laid out. Fourth, the experimental results and discussion of results are discussed and examined.

5.1 Data Sets

We used two different data sets that contain only short text documents. The characteristics of the two data sets are explained as the following:

- **Twitter.** We collected a corpus of tweets which contains 80,231 tweet posts. We collected tweets in the time frame from April 1, 2013 to April 30, 2013. We filter out all non-Latin characters tweets. Afterwards, we deleted any non-English tweets by classifying a tweet to be non-English if there is less than 5 English words. Moreover, we discarded any tweet with less than 3 words as it does not show any topic relevance. Moreover, the corpus contains 19,613 hashtags in total.

- **Opinosis.** We used a public short reviews data set called Opinosis[3] collected by Ganesan et al.[10]. The data set contains short reviews, a sentence long, about products collected from TripAdvisor, Amazon, and Edmunds. The data set contains 51 topics about a number of different products. For each topic, there is approximately a 100 short review snippet. A golden summary for each topic is created to summarize the reviews. There are 5 different golden summaries for each topic created by human workers from Amazon Mechanical Turk (MTurk)[4]. We randomly used 3 different topics to quantitatively test our algorithm with other baselines which are Windows7 features, iPod video, and Amazon's kindle price.

5.2 Preprocessing

Preprocessing is an essential step in text mining tasks in general. In extracting keyphrases, the preprocessing is needed to measure the salient scores accurately. The amount of preprocessing differs significantly depending on the genre of the corpus. In a social microblogging environment as in Twitter, the preprocessing step is of a vital importance. The challenge with colloquial textual content is an enormous obstacle in performing keyphrase extraction with Twitter posts. For instance, tweets can have misspelled words, strange capitalization, and wrong punctuations. For more detail we refer the reader to Eisenstein's survey on languages in social media [8]. Therefore, we did in an extensive preprocessing to tweets.

We first removed any URL links from tweets since we are focusing on the textual content. Moreover, we also removed emoticons and smileys since they do not have any topical relevance. Also, Twitter's special characters and usernames were removed as in the preceding hashtag sign # and usernames with @username. Tweets that start with the @username are generally considered replies and have a conversational nature more than topical nature. Therefore, we have removed any tweet that starts with @username to focus on topical tweets only. Another challenge is the usage of Internet phrasal abbreviation such as LOL (laugh out loud), ikr (I know right). We leverage the Internet Slang[5] dictionary in an effort to transform the text to standard English. All the aforementioned techniques can help improve the accuracy of the POS tagger.

Syntactical tagging, as in POS, for conversational content found on tweets can be very difficult. Most standard taggers fail to correctly tag colloquial text. For instance, the misuse of capitalization can make the tagger incorrectly tag nouns or verbs as a proper noun simply because the token is out of the vocabulary OOV. To tackle such difficulties, we have leveraged a state-of-the-art POS tagger[6] designed specifically for tweets [21]. The tagger designed at Carnegie Melon University is capable of accurately tagging tokens in a noisy genre as in Twitter. Moreover, the tagger is capable of identifying tags regardless of the capitalization misuse or the strange orthography of text, for example repeating letters for emphasis as in soooo. After tagging the tweets, we focused on selecting nouns and adjectives only since they are the base for noun phrases. We finally removed stopwords and stemmed the tokens.

The final step of preprocessing was to remove all stopwords from tweets since they do not have any topical influence. Punctuations were removed as well. Moreover, all capitalized tokens were converted to lower case. We lastly stemmed the tokens to get an accurate feature measure of words. We used the Porter stemmer[7] to stem our corpus.

For the Opinosis data set, we removed stopwords, punctuations, and stemmed the text. We also convert the tokens to lower case.

5.3 Experimental Setup

In this section, we will describe the experimental setup that was used for both data sets to compare our model with other baselines. First, we will describe the setup used for the Twitter data set. Then, we will explain the setup for the Opinosis review data.

Since there is no apparent golden labels to test against with tweets, we designed an empirical experiment to test keyphrase extraction in tweets. The experiment can be designed into different steps 1) Identify major topics in documents (tweets). 2) Test if any top ranked phrase represents a major topic in documents. The intuition behind the approach is the fact that phrases are descriptive of a document if they explain an important topic within that document. Given that keyphrases describe the major topics in a document, we will leverage a statistical topic model known as latent Dirichlet allocation (LDA)[4] to first extract the main topics in documents using a new twitter representation that improves the topic model with short documents. Second, we use search engines to search these topics to generate gold-label phrases. Finally, we test and compare all the ranking baselines by using the search results from the major topics in Twitter and golden summaries in Opinosis separately.

LDA is a generative statistical model that helps finding a set of unobserved groups using some observed sets. When applied to text, the observed sets are words in documents where the unobserved groups (latent) are topics of co-occurring words. By finding the mixture of topics using statistical inference as in Gibbes sampling, we get two posterior distribution $P(w|k)$ the probability of words under each topic, and $P(k|d)$ the probability of topics under each document. We first start by assigning each word to a K topic. Then, for each word w in each document d, we resample the

[3]http://kavita-ganesan.com/opinosis-opinion-dataset
[4]www.mturk.com
[5]http://www.noslang.com/dictionary/full/
[6]http://www.ark.cs.cmu.edu/TweetNLP/
[7]http://tartarus.org/martin/PorterStemmer/

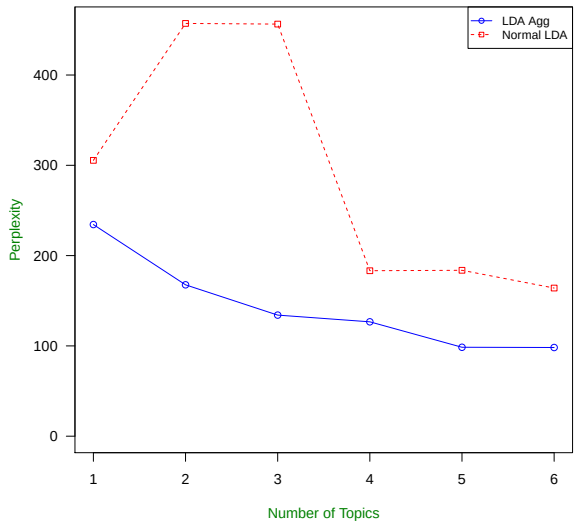

Figure 1: Comparison between LDA with Short Documents Compared to Aggregated Documents (↓)

probability distribution as follows:

$$P(w_i|d) = \sum_{s=1}^{|K|} P(w_i|k_s)P(k_s|d) \quad (15)$$

Where $P(w_i|k_s)$ is the probability of the word w_i being assigned topic k_s from all documents. $P(k_s|d)$ is the probability of words in the document d that are assigned to topic k_s. However, when applied to short length documents as in tweets, a new challenge arises since there is not enough observed sets (words) in each document to infer latent topics. Therefore, we remodel the documents structure to improve LDA in our data and get meaningful topics. Given a collection of tweets $\mathcal{T}_i = \{\tau_1, \tau_2, ..., \tau_n\}$, there is an abundant number of hashtags $\mathcal{H}_i = \{h_1, h_2, ..., h_n\}$ appearing in tweets. Instead of treating each tweet as a document, we aggregate tweets using hashtags to form a large virtual document for each hashtag $d_h = \{\tau_{1h} + \tau_{2h} + ... + \tau_{nh}\}$ where each d_h is a concatenation of tweets. Therefore, the documents set will be defined as $D_h = \{d_{h1}, d_{h2}, ..., d_{hn}\}$ containing all words from a large group of tweets for each document. After enhancing the document representation, we can apply LDA to learn topics and their posterior ranking more efficiently. In Figure 1, we show a significant improvement in perplexity for the two LDA approaches with short documents and aggregated documents. In Table 1, we show the top 10 ranked word for $|K| = 5$. Similar approaches for improving LDA in short text documents are found in [19].

To test the hypergraph ranking, we would need to have a reference set that summarize topics within each d_h document. The idea is to use a search engine by using the top words, from LDA, for each topic as a query. We used Google to generate the result snippets by setting the search at the same duration as the tweets which is April, 2013. Once the search snippets (top 50 snippet) for each topic is collected, we store them. Then we assign each document to its major topics only. For instance, any topic that is higher than $P(k|d) = 0.5$ is considered a major topic and is then chosen. Those search results collected from the major topics are considered references. We, then, search for keyphrases

Table 1: Top 10 Words of Five Topics

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
super	health	obama	gun	social
bowl	care	barack	control	media
jar	law	presid	peopl	market
utc	congress	agenda	stricter	dimens
box	favor	buchanan	grip	roi
fuse	news	cien	sens	twitter
look	work	con	great	amp
good	alli	cumpl	america	use
beyonc	amp	mandato	anti	current
black	job	congress	common	post

in those topical references. If a keyphrase is included in a snippet of any major topic, we consider it a hit, otherwise it is a miss.

For the Opinosis review data, we compared the extracted keyphrases from short review documents with the golden summaries provided with the data set. We consider a keyphrase correct if it appears in any short golden summary. There is approximately 5 golden summaries for each topic. Due to the short length of documents, we only used bigrams for evaluation.

A number of different baseline algorithms are implemented and used to test the validity of the proposed approach:

- **tf-idf** each post is a document, and each topic collection is a reference corpus.
- **TextRank**[20] builds a graph of keywords with sliding window $w = 2$. Edge weights are the frequency of co-occurring relation.
- **TimedTextRank**[24] builds a graph of keywords similar to TextRank. The ranking is, however, multiplied by a time function over the destination node.
- **NGTS(normal graph with Time and Social)** A normal lexical graph similar to TextRank. However, the edge weights are the summation of temporal and social function over all documents that contain the pair of words u and v (instead of frequency of co-occurrence)
- **NE-Rank**[3] A normal lexical graph similar to TextRank. However, the ranking takes into account node and edge weights. Node weights are tf-idf and edge weights are the frequency of co-occurrence. It also considers node weights when the random walk teleports to a random node.

5.4 Experimental Results & Discussion

To compare all the baselines used in this experiment, we quantitatively measure their performance using a precision evaluation metric. We compare them to the golden labels defined in the previous section. Specifically, we consider the keyphrase to be correct if it appears in the golden set. Precision helps identify the accuracy of the extracted results. Since we are evaluating a ranking system, we measure precision at the top 10, 15, and 20 ranked keyphrases. Precision is measured as follows:

$$Precision = \frac{K_{correct}}{K_{extracted}} \quad (16)$$

Table 2: Keyphrase Extraction Experimental Results for Twitter using Precision

	P@10	P@15	P@20
tf-idf	0.28	0.26	0.24
TextRank	0.52	0.39	0.28
TimedTextRank	0.54	0.424	0.32
NGTS	0.48	0.40	0.32
NE-Rank	0.56	0.426	0.32
HG-RANK	0.64	0.49	0.40

Table 3: Keyphrase Extraction Experimental Results for Opinosis using Precision

	P@10	P@15	P@20
tf-idf	0.36	0.28	0.28
TextRank	0.53	0.42	0.35
NE-Rank	0.60	0.46	0.36
HG-RANK	0.66	0.57	0.46

Where $K_{correct}$ is the number of correctly extracted keyphrase, and $K_{extracted}$ is the total number of extracted keyphrase. In the following, we will discuss the experimental results with balancing social and temporal attributes and how it affect the ranking. Then, we will describe the improvements HG-RANK has over other approaches.

However, precision only considers how many correctly extracted keyphrases within the result regardless of the order within the top list of extracted keyphrases. Therefore, we also measure the Binary Preference Measure Bpref [7]. Bpref will penalize the system if incorrect keyphrases ranked higher than correct keyphrases. Bpref is measured as the following:

$$Bpref = \frac{1}{R}\sum_{r\in R} 1 - \frac{|\text{n } \textit{ranked higher than} \text{ r}|}{R} \qquad (17)$$

where R is the number of correct keywords within extracted keywords in a method, and where r is a correct keyword and n is incorrect keyword.

To examine the effect of social and temporal attributes when combined into the hypergraph ranking scheme, we experimented with different values for λ in formula 6. By varying the value of λ, we can analyze the tradeoff between the two attributes in precision. Figure 2 shows the different λ values experimented with. The best value for λ is 0.5 which means equal contribution of temporal and social features in our data. It is interesting to notice that when $\lambda = 1$, meaning only social attributes were taken into the ranking, the performance deteriorates considerably. It could mean that popular content is not necessary of a topical value to the corpus. However, more experiments are needed to widen our understanding of what the best features are for topical keyphrase extraction in a dynamic genre as Twitter. Next, we move on to describe the full evaluation of both data sets.

To evaluate the ranking performance for all baselines, we performed the evaluation measure for both data sets as the following. For Twitter, we first build the lexical hypergraph for each hashtag topic corpus $d_h = \{\tau_{1h} + \tau_{2h} + ... + \tau_{nh}\}$. We chose the top 5 frequent topical hashtags and performed keyphrase extraction separately. We measured the precision and Bpref for each topic. In table 2, we show the average precision from all topics. Table 4 shows the average Bpref for the Twitter data. In the Opinosis review data set, we build

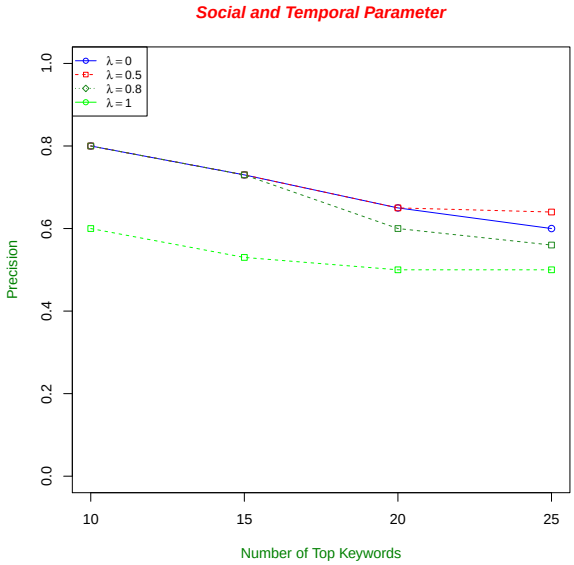

Figure 2: Different Effect between Temporal & Social Attributes

Table 4: Keyphrase Extraction Experimental Results for Twitter using Bpref

	Bpref@10	Bpref@15	Bpref@20
tf-idf	0.66	0.42	0.40
TextRank	0.744	0.74	0.74
TimedTextRank	0.748	0.67	0.67
NGTS	0.742	0.71	0.71
NE-Rank	0.76	0.75	0.75
HG-RANK	0.82	0.82	0.82

the lexical hypergraph for each topic. Since there is no meta data with the reviews as time or social features, we regard the hyperedge weight to be 1 for all short documents to test the hypergraph ranking only. We chose 3 topics mentioned early to quantitatively measure the improvements. We show the average precision of three topics in table 3, and average Bpref in table 5.

In the Twitter data, the proposed hypergraph-based approach HG-RANK out performed all other baselines. Specifically, HG-RANK improved the results in the top 10 results over closest baseline, NE-Rank, by 14% and 7% improvements using precision and Bpref, respectively. The improvement shows the importance of modeling the high-order co-occurring relationship using a lexical hypergraph compared to modeling just a pair of words for graph edges. Moreover, the temporally-aware ranking HG-RANK showed improvement over other temporal-aware approaches as in TimedTextRank[24] and NGTS. Similar improvements are demonstrated for the top 15 and top 20 keyphrases.

For the Opinosis data, HG-RANK showed improvement over all baselines as well. Improvements in the top 10 over the second best baseline, NE-Rank, were 10% in precision and 14% in Bpref. Moreover, similar improvements were found in the top 15 and top 20 keyphrases. Even though no hyperedge weights were used for this data set as in temporal and social attributes, the hypergraph model has shown to increase both precision and Bpref scores which shows the robustness of the proposed model in modeling high-order co-occurrence relation between words.

Table 5: Keyphrase Extraction Experimental Results for Opinosis using Bpref

	Bpref@10	Bpref@15	Bpref@20
tf-idf	0.61	0.61	0.61
TextRank	0.88	0.78	0.78
NE-Rank	0.82	0.80	0.80
HG-RANK	0.94	0.82	0.82

6. CONCLUSION

In this paper, we have proposed a hypergraph-based ranking algorithm suitable for short text documents in social media genre. We modeled distinct keywords as vertices and their short documents as hyperedges in a lexical hypergraph. Moreover, we jointly modeled temporal and social features in the hypergraph to adapt keyphrase extraction with the dynamic nature of social media. Additionally, we supplemented the hypergraph with discriminative weights in the vertices to enhance the random walk approach. Then we proposed a new probabilistic random walk that considers both vertices and hyperedges weights over hypergraph. We have leveraged a state-of-the-art POS tagger for Twitter data to capture syntactic tags accurately from the noisy text. We demonstrated the effectiveness of our hypergraph approach over two data sets which showed promising results.

In the future work, we plan to extend the approach to a streaming algorithm where the hypergraph can be updated periodically.

7. REFERENCES

[1] C. Avin, Y. Lando, and Z. Lotker. Radio cover time in hyper-graphs. In *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*, DIALM-POMC '10, pages 3–12, New York, NY, USA, 2010. ACM.

[2] A. Bellaachia and M. Al-Dhelaan. Learning from twitter hashtags: Leveraging proximate tags to enhance graph-based keyphrase extraction. In *Proceedings of the 2012 IEEE GreenCom*, pages 348–357, Washington, DC, USA, 2012. IEEE Computer Society.

[3] A. Bellaachia and M. Al-Dhelaan. Ne-rank: A novel graph-based keyphrase extraction in twitter. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence*, WI-IAT '12, pages 372–379, Washington, DC, USA, 2012. IEEE Computer Society.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, Mar. 2003.

[5] A. Bougouin, F. Boudin, and B. Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth IJCNLP*, pages 543–551, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[7] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR*, pages 25–32, New York, NY, USA, 2004. ACM.

[8] J. Eisenstein. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the NAACL*, pages 359–369, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[9] G. Erkan and D. R. Radev. Lexrank: graph-based lexical centrality as salience in summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, Dec. 2004.

[10] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd COLING*, pages 340–348, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[11] Y. Gao, J. Liu, and P. Ma. The hot keyphrase extraction based on tf*pdf. In *The 2011 IEEE 10th TrustCom*, pages 1524–1528, 2011.

[12] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In M. Collins and M. Steedman, editors, *Proceedings of the 2003 EMNLP*, pages 216–223, 2003.

[13] L. Jabeur, L. Tamine, and M. Boughanem. Featured tweet search: Modeling time and social influence for microblog retrieval. In *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence*, volume 1, pages 166–173, 2012.

[14] D. Li and S. Li. Hypergraph-based inductive learning for generating implicit key phrases. In *Proceedings of the 20th WWW*, pages 77–78, New York, NY, USA, 2011. ACM.

[15] D. Li, S. Li, W. Li, W. Wang, and W. Qu. A semi-supervised key phrase extraction approach: Learning from title phrases through a document semantic network. In *Proceedings of the ACL*, pages 296–300, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[16] X. Li, B. Liu, and P. Yu. Time sensitive ranking with application to publication search. In *The Eighth IEEE ICDM*, pages 893–898, 2008.

[17] X. Liu, Y. Li, F. Wei, and M. Zhou. Graph-based multi-tweet summarization using social signals. In *Proceedings of COLING*, pages 1699–1714, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[18] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 EMNLP*, pages 366–376. Association for Computational Linguistics, October 2010.

[19] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international ACM SIGIR*, pages 889–892, New York, NY, USA, 2013. ACM.

[20] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In D. Lin and D. Wu, editors, *Proceedings of the 2004 EMNLP*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

[21] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the NAACL*, pages 380–390, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[22] Z. Ren, S. Liang, E. Meij, and M. de Rijke. Personalized time-aware tweets summarization. In *Proceedings of the 36th International ACM SIGIR*, pages 513–522, New York, NY, USA, 2013. ACM.

[23] R. Sipos, A. Swaminathan, P. Shivaswamy, and T. Joachims. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM CIKM*, pages 754–763, New York, NY, USA, 2012. ACM.

[24] X. Wan. Timedtextrank: adding the temporal dimension to multi-document summarization. In *Proceedings of the 30th annual international ACM SIGIR*, pages 867–868, New York, NY, USA, 2007. ACM.

[25] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial intelligence - Volume 2*, pages 855–860. AAAI Press, 2008.

[26] W. Wang, F. Wei, W. Li, and S. Li. Hypersum: hypergraph based semi-supervised sentence ranking for query-oriented summarization. In *Proceedings of the 18th ACM CIKM*, pages 1855–1858, New York, NY, USA, 2009. ACM.

[27] P. S. Yu, X. Li, and B. Liu. Adding the temporal dimension to search " a case study in publication search. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 543–549, Washington, DC, USA, 2005. IEEE Computer Society.

[28] X. Zhao, J. Jiang, J. He, Y. Song, P. Achanauparp, E.-P. Lim, and X. Li. Topical keyphrase extraction from twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 379–388, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[29] D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. *NIPS*, 19:1601, 2007.