Tweet Timeline Generation via Graph-based Dynamic Greedy Clustering

Feifan Fan[1], Runwei Qiang[1], Chao Lv[1], Wayne Xin Zhao[2] and Jianwu Yang[1,*]

[1] Institute of Computer Science and Technology
Peking University, Beijing 100871,China
{fanff, qiangrw, lvchao, yangjw}@pku.edu.cn
[2] School of Information, Renmin University of China
batmanfly@gmail.com

Abstract. When searching a query in the microblogging, a user would typically receive an archive of tweets as part of a retrospective piece on the impact of social media. For ease of understanding the retrieved tweets, it is useful to produce a summarized timeline about a given topic. However, tweet timeline generation is quite challenging due to the noisy and temporal characteristics of microblogs. In this paper, we propose a graph-based dynamic greedy clustering approach, which considers the *coverage*, *relevance* and *novelty* of the tweet timeline. First, tweet embedding representation is learned in order to construct the tweet semantic graph. Based on the graph, we estimate the *coverage* of timeline according to the graph connectivity. Furthermore, we integrate a noise tweet elimination component to remove noisy tweets with the lexical and semantic features based on *relevance* and *novelty*. Experimental results on public Text Retrieval Conference (TREC) Twitter corpora demonstrate the effectiveness of the proposed approach.

Keywords: Tweet Timeline Generation; Graph-based Dynamic Greedy Clustering; Tweet Embedding;

1 Introduction

Microblogging has become one of the most popular social networking platforms in recent years. When users search a query in a microblogging service such as Twitter, an archive of tweets would be returned as part of a retrospective piece on the impact of social media on a specific topic. For instance, a journalist may invest a sports scandal that has been brewing for the past several weeks. She just got news of a breaking development, and turns to searching tweets to find more details. However, due to the retweeting and sharing nature of Twitter, the traditional search engine would lead to a lot of duplicates or near-duplicates tweets that contain the same or highly-similar information - a user cannot easily get an overall idea of the retrieved these tweets. Thus, it would be helpful if the search system produced a “summary” timeline about the topic, which is the studied task in this paper.

In TREC 2014 Microblog track, the organizer introduced a novel pilot task named Tweet Timeline Generation (TTG) task [11]. The TTG task can be summarized as “*At*

*Corresponding author.

time T, I have an information need expressed by query Q, and I would like a summary that captures relevant information". Developing effective TTG system is inherently challenging. Aside from the challenges derived from the tweet retrieval with issues from topic detection and tracking (TDT) and traditional multi-document summarization, systems should further address three additional challenges. First of all, as the length of microblog entry is limited to 140 characters and the content of tweet can be very noisy, it is hard to detect redundant tweets (or cluster similar tweets) using traditional bag-of-words representation of tweets. Generally, users are more interested with tweets which are highly relevant with the query, and a good summarized timeline should be concise and contain as few redundant tweets as possible. Secondly, topics evolve quickly in social media, people usually talk about a subtopic of a given topic in a specific time period. Hence, to measure the similarity of tweets, systems should also take the temporal information into consideration. Thirdly, different topics can attract different amount of attention, which leads to different amount of relevant tweets. As in the search scenario of TTG, users are assumed ready to consume the entire summarized tweet list (unlike a ranked list). Therefore, it is quite necessary for the system to keep the *coverage* of raw related tweet collection in the summarized timeline.

In this work, we mainly address the above challenges (beyond tweet retrieval) in TTG and propose a graph-based dynamic greedy clustering approach to characterize the *coverage*, *relevance* and *novelty* properties of the tweet timeline. The major contributions of this work are: (1) We propose to learn tweet embedding representation to characterize the similarity between tweets which considers both the semantic relatedness and time proximity. We further utilize the similarity to construct tweet semantic graph. (2) We propose a dynamic greedy clustering approach based on tweet semantic graph, where we estimate the *coverage* according to the vertex connectivity in the graph and we integrate a noise tweet elimination component based on logistic regression classifier to measure the *relevance* and *novelty* using many effective lexical and semantic features. (3) We construct extensive experiments on public Text Retrieval Conference (TREC) Twitter corpora, which demonstrate the effectiveness of the proposed approach.

The remainder of the paper is organized as follows. Section 2 introduces related work on subtopic retrieval, TDT, and timeline generation. The graph-based dynamic greedy clustering approach is presented in Section 3. The experimental results as well as the comparisons with the-state-of-arts are shown in Section 4. Finally, we conclude the paper and outline our future work in Section 5.

2 Related Work

Subtopic Retrieval Zhai *et al.* [19] presented *subtopic retrieval problem*, which is concerned with finding documents that cover many different subtopics of a given topic. Subtopic retrieval is quite different from traditional retrieval problem, where the search engines just simply return the search results in general. However, the retrieved documents always contain much redundant or noisy information in reality. Agrawal *et al.* [2] proposed a systematic approach to diversify the searched results, which tries to maximize the likelihood of selecting a relevant document in the top-k positions based on the categorical information of the queries and documents. Marco *et al.* [5] employed a method relying on n-grams to cluster and diversify web search results. They constructed a co-occurrence graph based on Dice coefficient calculated over the corpus in which the senses are discovered by word sense induction algorithm. In this way, the method can better capture the similarity between the web snippets. To build a more realistic and efficient solution, which allows to label for subtopics or aspects of a given query, Wang *et al.* [17] adopted the star clustering algorithm proposed in [4].

Unlike subtopic retrieval problem, research in the area of TTG aims to obtain a sequence of documents that could describe how a topic evolves over time. In other word, the temporal information must be incorporated into the TTG system.

Topic detection and tracking TDT task mainly conveys the recognition and evolution of the topics contained in text streams. Many previous works [7, 9] detect topic through discovering topic bursts from a document stream. Among them, detecting the frequency peaks of topic-related phrases over time in a histogram is a common solution. Another main technique attempted to monitor the formation of a cluster from a structure perspective. Lappas *et al.* [7] presented a approach to model the burstiness of a term, using discrepancy theory concepts. They could identify the time intervals of maximum burstiness for a given term, which is an effective mechanism to address topic detection in the context of text stream. Lin *et al.* [9] proposed burst period detection based on the phenomenon that at some time point, the topic-related terms should appear more frequently than usual and should be continuously frequent around the time point. Agarwal *et al.* [1] discovered events as dense clusters in highly dynamic graphs. Following the same idea, Lee *et al.* [8] applied a clustering algorithm named DBSCAN to recognize the evolution pattern of a topic.

Though these methods have been successfully adopted in TDT task, they are not applicable to the TTG problem. The timeline generation problem represents a natural extension of traditional retrieval [11], which means the generation process is based on the documents returned by the search engines. Therefore, major techniques used in TDT such as burst period detection and dense-based clustering cannot be well applied in generating timeline since many subtopics or aspects in the timeline just contain exactly one document.

Timeline Generation There are also several works studying the timeline generation recently. A greedy algorithm based on approximation of Minimum-Weight Dominating Set Problem (MWDS) is exploited in [16, 9, 21]. Among these works, Wang *et al.* [16] proposed an approach that combines image and text analysis to generate a timeline containing textual, pictorial and structural information. They first constructed a multi-view graph, in which each node contains textual and pictorial information, and then selected the representative nodes by finding a minimum dominant set on the graph. Based on the same idea, Lin *et al.* [9] adopted the method to tweet timeline generation. Xu *et al.* [18] proposed a novel detection approach, framing the problem of redundant tweet removal as a sequential binary decision task. Lv *et al.* [12] applied hierarchical clustering algorithm based on Euclidean distance and adaptive relevance estimation to generate tweet timeline, which achieved the best performance of TTG task in TREC 2014 Microblog Track.

These methods had acquired decent effect on timeline generation, while they didn't well handle the unique characteristics of microblog, especially the *coverage*, *relevance* and *novelty* of the timeline. In this paper, we propose a graph-based dynamic greedy clustering approach, we first construct tweet semantic graph using tweet embedding representation, which considers both the semantic relatedness and time proximity in a more comprehensive way. Based on the graph, we estimate the *coverage* of timeline according to the vertex connectivity in the graph, and measure the *relevance* and *novelty* through noise tweet elimination component which utilizes many effective lexical and semantic features.

3 The Proposed Approach

In this section, we first introduce the problem formulation, and then present our approach. The proposed approach first constructs tweet semantic graph based on tweet embedding representation, and then apply graph-based dynamic greedy clustering algorithm to generate the summarized tweet timeline, where we integrate a detection component to eliminate noisy tweets.

3.1 Problem Formulation

We give a formal definition of TTG as follows:

Input: Given a topic query $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$ from users, where q_i is a query term, we obtain a tweet collection $C = \{T_1, T_2, \cdots, T_N\}$ related to the query by traditional retrieval model, where T_i is a tweet and N is the number of retrieved tweets.

Output: A summarized tweet timeline which consists of relevant and non-redundant, chronologically ordered tweets, i.e. $R^{(Q)} = \{T_1^{(Q)}, T_2^{(Q)}, \cdots, T_K^{(Q)}\}$, where $T_i^{(Q)}$ is a relevant tweet from C for query Q, and K is the number of tweets in the timeline.

3.2 Tweet Semantic Graph Construction

Definition 1. *(TWEET SEMANTIC GRAPH): A tweet semantic graph $G = (V, E)$, where V is a set of tweet vertices and E is a set of undirected edges, which represents the semantic relatedness between tweets.*

It is infeasible and meaningless to consider all the tweets for query-specific timeline generation. We apply state-of-the-art retrieval model to derive the top 300 ranked tweets as candidate. Given a candidate tweet collection with their timestamps, we construct a tweet semantic graph by viewing the tweets as the vertices V and calculating the weights of undirected edges on the basis of both content similarity and time proximity. Let T_i and T_j be two vertices in V. We make a undirected edge between T_i and T_j if and only if the similarity between them is greater than a similarity threshold σ.

Tweet Embedding Representation Due to the length limit and informal expression of tweets, traditional retrieval models relying on "bag-of-words" representations are faced with challenges in describing the similarity of short tweets. For example, when talking about "happy birthday" in twitter, users may use many informal words, such as "bday" and "birthdayyy". Besides, users may use different words to express same meaning, such as "word shortage" and "drought", while these words are semantically related from the context of the twitter stream.

Inspired by distributed representation methods [13], we propose to learn tweet embedding representation which can project tweets into low-dimensional semantic space. Give a tweet $T = \{w_1, w_2, \cdots, w_{|T|}\}$, the objective function is to maximize the log-likelihood, defined as

$$\mathcal{L} = \frac{1}{|T|} \sum_{i=1}^{|T|} \log Pr(w_i | w_{i-c} : w_{i+c}) \tag{1}$$

where $w_{i-c} : w_{i+c}$ is the subsequence $(w_{i-c}, \ldots, w_{i+c})$ by excluding w_i and $2 \times c$ is the window size. We model each word w_i an M-dimensional embedding vector $\boldsymbol{v}_{w_i}$. With this form of embedding representation, we further employ a multiclass classifier softmax to formulate the probability $Pr(w_i | w_{i-c} : w_{i+c})$ as follows

$$Pr(w_i | w_{i-c} : w_{i+c}) = \frac{\exp(\bar{\boldsymbol{v}}^\top \boldsymbol{v}'_{w_i})}{\sum_{w \in \mathcal{W}} \exp(\bar{\boldsymbol{v}}^\top \boldsymbol{v}'_{w})} \tag{2}$$

where $\mathcal{W}$ is the word vocabulary, $\boldsymbol{v}'_{w_i}$ is the output vector representation of target word w_i and $\bar{\boldsymbol{v}}$ is the averaged vector representation of the context. We simply average all the word vectors to obtain the tweet vector. Based on the tweet vector, we can utilize classic similarity measure (i.e. cosine similarity) to estimate the similarity of tweets. Considering that tweets posted in a same time interval (e.g. the latest two hours) are more likely to talk about the same aspect of the topic, thus we combine the tweet vector with fading time factor to characterize both the semantic relatedness and time proximity in a more comprehensive way, which is defined as follows

$$sim(T_i, T_j) = \frac{|\boldsymbol{v}_i \cdot \boldsymbol{v}_j|}{|\boldsymbol{v}_i| \cdot |\boldsymbol{v}_j|} \cdot e^{-\gamma |\tau_i - \tau_j|} \tag{3}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are the tweet embedding representations of T_i and T_j, respectively. γ is the exponential parameter that controls the temporal influence. τ_i and τ_j are the corresponding timestamps, measured in fractions of hours.

3.3 Graph-based Dynamic Greedy Clustering Approach

As discussed in Section 1, In order to obtain the summarized tweet timeline from retrieved tweets, the system must dynamically detect and eliminate the redundant or noisy tweets for distinct topics. We propose a dynamic greedy clustering approach based on tweet semantic graph, which considers both the semantic relatedness and time proximity between tweets. Considering that the returned tweets from the retrieval model can still contain much noise, we further incorporate a noise elimination component based on both lexical and semantic similarity during the clustering process.

Graph-based Dynamic Greedy Clustering Algorithm The proposed algorithm iteratively identify representative vertices given a query, where *relevance*, *coverage* and *novelty* have been considered. We present the overall procedure in Algorithm 1.

Given a set of retrieved tweets as the input, at Line 1 we construct the tweet semantic graph $G = (V, E)$. At the beginning of each iteration, we first (Line $3 \sim 7$) compute the *coverage* score for each remaining (unmarked) vertex by using a score function $ComputeCoverageScore(v_i)$ that calculates the connectivity based on unmarked vertices connected with v_i in the graph. Besides *coverage*, users would be more interested in tweets which are highly relevant with the query, and a good summarized timeline should be concise and contain as few redundant tweets as possible. In order to capture the *relevance* and *novelty* for timeline generation, we check the top ranked tweet (Line $10 \sim 16$) by using a boolean function $IsNoiseTweet(\cdot)$ based on a rich set of relevance and novelty features, which will be discussed next. We perform the update procedure for marking as follows. If a tweet is measured as *noise* by $IsNoiseTweet(\cdot)$, the corresponding vertex is marked as *noise vertex*. Otherwise, the corresponding vertex is marked as the *centroid vertex* which will be incorporated into the timeline $R^{(Q)}$. The rest neighboring vertices of the centroid vertex are marked as *visited* and no more considered for selection.

Algorithm 1 Graph-based Dynamic Greedy Clustering.

```
Input:  Candidate tweet collection $C = \{T_1, T_2, \cdots, T_N\}$
    Query $Q$
    Edge similarity threshold $\sigma$
    The maximum number of tweets in timeline $K$
    Exponential parameter $\gamma$ in Eq. 3
Output:
    Summarized tweet timeline $R^{(Q)} = \{T_1^{(Q)}, T_2^{(Q)}, \cdots, T_K^{(Q)}\}$
 1: $G(V, E) \leftarrow ConstructGraph(C, \sigma, \gamma)$
 2: repeat
 3:     for $v \in V$ do
 4:         if $v_i$ is not marked then
 5:             $ComputeCoverageScore(v_i)$;
 6:         end if
 7:     end for
 8:     Rank the remaining vertices by the coverage scores;
 9:     $v^* \leftarrow$ top ranked vertex;
10:     if $IsNoiseTweet(v^*, Q, R^{(Q)})$ then
11:         Mark $v^*$ as a noise vertex;
12:     else
13:         Mark $v^*$ as the centroid vertex;
14:         Mark the vertices connected to $v^*$ as visited;
15:         $R^{(Q)} \leftarrow R^{(Q)} \cup \{v^*\}$;
16:     end if
17: until $\forall v \in V$ is marked or $|R^{(Q)}| = K$
18: return $R^{(Q)}$
```

Noise Tweet Elimination In Algorithm 1, a key component is the noise elimination function $IsNoiseTweet(\cdot)$, which aims to filter noise tweets and improve the timeline quality. To implement $IsNoiseTweet(\cdot)$, we utilize a logistic regression classifier based on a rich set of lexical and semantic features to eliminate noise tweet, which have been used to achieve the state-of-art in adaptive filtering problem of news and tweets [20, 3]. These two types of features are given as follows.

- **Relevance Features** measure the relevance between tweet T and query Q. We have four relevance features in total. Based on traditional "bag-of-words" model, three lexical similarity score features are calculated by cosine similarity, Dice coefficient, and Jaccard coefficient, respectively. In addition, one semantic similarity score feature is obtained by cosine similarity measure using tweet embedding representation.
- **Novelty Features** estimate the novelty of an unprocessed tweet compared with previous generated centroid tweets of clusters. We calculate the similarity between the unprocessed tweet and each centroid tweet of each generated cluster, and choose the closest centroid tweet as comparison to estimate the novelty of the unprocessed tweet. Like *Relevance Features*, we obtain three lexical and one semantic score features based on "bag-of-words" model and tweet embedding representation, respectively.

To train the logistic regression classifier, we use the judgments on TREC2011-2012 topics as labeled data, which are released and labeled by the official TREC organizer.

In summary, *coverage* is measured by using the function $ComputeCoverageScore(\cdot)$ that calculates the connectivity based on unmarked vertices in the graph, , while *relevance* and *novelty* have been considered in function $IsNoiseTweet(\cdot)$ by leveraging the relevance and novelty features. Our clustering algorithm can be efficiently implemented based on the well-known Breadth-First-Search (BFS) graph traverse algorithm.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of the proposed method. In what follows, we first describe the experimental setting and then present the results.

4.1 Experimental Setting

Dataset Two large data collections (i.e. Tweets2011 and Tweets2013 collections) are used in our experiments. TREC organizers release a streaming API to participants [10]. Using the official API [3], we crawled a set of local copies of the canonical corpora. Tweets11 collection has a sample of about 16 million tweets, ranging from January 24, 2011 to February 8, 2011 while Tweets13 collection contains about 259 million tweets, ranging from February 1, 2013 to March 31, 2013 (inclusive). Tweets11 is used for evaluating the effectiveness of the proposed Twitter TTG systems over 10 training topics in TREC 2011 and 2012. And, Tweets13 is used in evaluating the proposed TTG

[3] https://github.com/lintool/twitter-tools

systems over 55 official topics in the TREC 2014 Microblog track [11]. The topics of TREC 2011-2012 are used for tuning the parameters and then we use the best parameter setting to evaluate our methods with topics for TREC 2014. Table 1 summarizes basic statistics of the two corpora.

Table 1. Data statistics of Tweets2011 and Tweets2013 test collections. K is the average number of tweets in the gold timelines.

Corpus	#queries	#tweets	#tokens	#terms	K
Tweets2011	10	4,948,137	753,021	27,180,607	130
Tweets2013	55	68,682,325	5,063,852	298,321,176	193

Evaluation Metrics Our evaluation metrics contain the following two types of metrics.

Tweet retrieval: As our TTG system's input is a candidate tweet collection generated by retrieval models, the performance of our system might be affected by the retrieval performance. In TREC Microblog track, tweets are judged on the basis of the defined information using a three-point scale [14]: *irrelevant* (labeled as 0), *minimally-relevant* (labeled as 1), and *highly-relevant* (labeled as 2). We use two official main metrics for the retrieval task in TREC, including Mean Average Precision (MAP) and Precision at N (P@N). Specifically, MAP for top 1000 ranked documents and P@30 with respect to *allrel* (i.e. tweet set labeled as 1 or 2).

Clustering performance as timeline quality: TTG results will be evaluated by two different versions of the F_1 metric, i.e., an unweighted version and a weighted version, which are used in TREC 2014 Microblog Track [11]. F_1 metric is combined by cluster precision and cluster recall. We first introduce the unweighted version as follows.

– **Cluster precision (unweighted).** Of tweets returned by the system, how many distinct semantic clusters are represented.
– **Cluster recall (unweighted).** Of the semantic clusters discovered by the assessor, how many are represented in the system's output.

For unweighted version, the system does not get "credit" for retrieving multiple tweets from the same semantic cluster. Different from unweighted F_1, the weighted F_1 (denoted as F_1^w) attempts to account for the fact that some semantic clusters are intuitively more important than others. Each cluster will be weighted by relevance grade: minimally-relevant tweets get a weight of one and highly-relevant tweets get a weight of two. These weights are then factored into the precision and recall computations. The F_1^w score is the main evaluation metric for TTG task in TREC 2014.

4.2 Methods to Compare

We consider the following methods as comparisons in our experiments.

– **TTGPKUICST2**: Hierarchical clustering algorithm based on adaptive relevance estimation and Euclidean distance, proposed in [12], which achieved the best performance in TREC 2014 Microblog Track.

- **EM50**: kNN clustering approach applied in [15], using a modified Jaccard coefficient (i.e. EM) and used top K retrieved results as candidates for clustering, which won the second place in TREC 2014 Microblog Track.
- **hltcoeTTG1**: The novel detection approach proposed by Tan *et al.* [18]. Unlike clustering methods, they framed the problem of tweet timeline generation as a sequential binary decision task. Therefore, they proposed a binary classifier to determine whether a coming tweet is novel and then compose the novel tweets as the summarized tweet timeline, which won the third place in TREC 2014 Microblog Track.
- **MWDSA**: we implement the Greedy MWDS Approximation Algorithm (denoted as **MWDSA**) that was exploited in generating storyline problem [16, 9, 21]. They identified the minimum-weight dominating set approximation as the most representative summary.
- **GDGC-BOW**: The proposed graph-based dynamic greedy clustering approach, which only utilizes "bag-of-words" model in both tweet graph construction and noise tweet elimination.
- **GDGC**: The proposed graph-based dynamic greedy clustering approach in Section 3.3.

Note that both in re-implemented systems and the proposed system, we obtain the top-ranked 300 tweets from the ranked list achieved by the retrieval models as the candidates for TTG process. We set the time factor γ as 0.01, usually around $0.005 \sim 0.02$. In tweet graph construction based on "bag-of-words" representation, we set the similarity threshold σ as 0.65, usually around $0.6 \sim 0.7$.

4.3 Results and Analysis

Recall that our TTG system's input is a candidate tweet collection generated by a retrieval model. In our study, we utilize two retrieval models for the candidate generation, namely **RTRM** and **RankSVM**. **RTRM** utilizes a two-stage pseudo-relevance feedback query expansion to estimate the query language model and expand documents with shortened URLs in microblog. In addition, **RTRM** can evaluate the temporal aspects of documents with the temporal-reranking components. **RankSVM** is a state-of-the-art pairwise learning to rank method, proposed in [6]. We follow the work of [12] and generate totally 250 features.

Table 2 presents the experimental results of TTG performance along with the candidate tweet retrieval performance. We can have the following obervations.

1) Compared with greedy **MWDSA** algorithm, **GDGC**-based methods outperform significantly in terms of F_1 and F_1^w for both **RTRM** and **RankSVM** retrieval candidate tweet collections. Since **MWDSA** considers the *coverage* by choosing the dominating vertexs in the graph, while it does not capture the *relevance* and *novelty* of tweet timeline. Besides, from the comparisons between two retrieval candidates, (e.g. **MWDSA**$_{RTRM}$ and **MWDSA**$_{RankSVM}$), we can also observe that TTG performance will benefit from better retrieval performance, which is very reasonable since TTG utilizes retrieval tweets as input candidates.

Table 2. Performance comparisons of the proposed methods and baselines. † and ‡ indicate that the corresponding improvements over **MWDSA**$_{RTRM}$ and **MWDSA**$_{RankSVM}$, are statistically significant ($p < 0.05$), respectively.

Method	Retrieval		TTG	
	MAP	P@30	F_1	$F_1{}^w$
TTGPKUICST2	0.5863	0.7224	0.3540	0.4575
EM50	0.5122	0.6982	0.2546	0.3815
hltcoeTTG1	0.5707	0.7121	0.2760	0.3702
MWDSA$_{RTRM}$	0.5422	0.6958	0.2581	0.3890
GDGC-BOW$_{RTRM}$	0.5422	0.6958	0.3364†	0.4295†
GDGC$_{RTRM}$	0.5422	0.6958	0.3468†	0.4452†
MWDSA$_{RankSVM}$	0.5863	0.7224	0.3143	0.4161
GDGC-BOW$_{RankSVM}$	0.5863	0.7224	0.3498‡	0.4556‡
GDGC$_{RankSVM}$	0.5863	0.7224	**0.3648**‡	**0.4754**‡

2) **GDGC** are consistently better than **GDGC-BOW**, which shows the importance of tweet embedding representation in constructing tweet semantic graph and estimating the *relevance* and *novelty* in noise tweet elimination component.

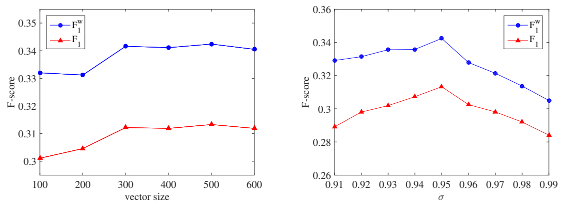
3) The proposed **GDGC**$_{RankSVM}$ outperforms best three systems in TREC 2014 Microblog Track, which demonstrates the effectiveness of the proposed approach in depicting the characteristics of tweet timeline. Besides, compared with **hltcoeTTG1**, the proposed approach using weaker retrieval results (i.e. **GDGC**$_{RTRM}$) can also perform better in TTG. Specially, our method **GDGC**$_{RTRM}$ improves the F_1^w score over **EM50** and **hltcoeTTG1** by 16.7% and 20.3%, respectively; while the corresponding increments in terms of F_1 are 36.2% and 25.7%. On the other hand, when utilizing a more effective retrieval model (i.e. **RankSVM**), the graph-based dynamic greedy clustering approach will achieve more improvements in terms of F_1^w and F_1. In addition, compared with **TTGPKUICST2**, **GDGC**$_{RankSVM}$ achieves 3.91% and 3.05% further increases in terms of F_1^w and F_1, respectively.

4.4 Parameter Tuning

Several parameters in the proposed method may affect the system performance. In this section, we analyze the parameter setting in the graph-based dynamic greedy clustering approach. All these experiments are run on TREC 2011-2012 topics.

Figure 1(a) shows the effect of tweet embedding vector in terms of metrics F_1^w and F_1. We can see that the two curves follow similar patterns, F-score increases rapidly with the increase of the embedding vector size when it is less than 300. When the vector size becomes larger, the performance changes slightly, which means that the vectors can already provide enough information to depict the tweets semantically from the contexts.

We study the effect of similarity threshold parameter σ in tweet semantic graph construction using tweet embedding representations, which is shown in Figure 1(b). we can observe that the value of σ yields a significant effect on the evaluation metrics of TTG (i.e. F_1^w and F_1).



(a) Effect of vector size. (b) Effect of threshold σ.

Fig. 1. Parameter Tuning.

Moreover, we can see that the optimal σ using tweet embedding representation is greater than that using "bag-of-words" model described in Section 4.2, which demonstrates the characteristics of tweet embedding representations. That is, words used in similar contexts are considered semantically similar and tend to have similar vectors, which could lead to a general high similarity score for tweets. When it comes to the "bag-of-words" representation, different words are simply regarded as irrelevant.

5 Conclusion and Future Work

In this study, we propose a graph-based dynamic greedy clustering approach. We utilize learned tweet embedding representation to construct tweet semantic graph, which considers both the semantic relatedness and time proximity in a more comprehensive way. Based on the graph, we estimate the *coverage* by highly scoring vertices with larger graph connectivity. For top ranked candidate tweet at each iteration, we measure the *relevance* and *novelty* through a logistic regression classifier which adopts many effective lexical and semantic features. Extensive experiments using public TREC Twitter collection, demonstrate the effectiveness of the proposed method.

Currently, we simply utilize the top-ranked 300 tweets from retrieval models as input candidates. In fact, the number of strong candidate tweets for distinct topics can be different due to the diverse popularity in Twitter. In the future, we will consider how to dynamically obtain candidate tweets from retrieval results.

Acknowledgments. The work reported in this paper is supported by the National Natural Science Foundation of China Grant 61370116. We thank anonymous reviewers for their beneficial comments.

References

1. Agarwal, M.K., Ramamritham, K., Bhide, M.: Real time discovery of dense clusters in highly dynamic graphs: identifying real world events in highly dynamic environments. Proceedings of the VLDB Endowment 5(10), 980–991 (2012)
2. Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S.: Diversifying search results. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining. pp. 5–14. ACM (2009)

3. Albakour, M., Macdonald, C., Ounis, I., et al.: On sparsity and drift for effective real-time filtering in microblogs. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. pp. 419–428. ACM (2013)
4. Aslam, J.A., Pelekhov, E., Rus, D.: The star clustering algorithm for static and dynamic information organization. J. Graph Algorithms Appl. 8, 95–129 (2004)
5. Di Marco, A., Navigli, R.: Clustering and diversifying web search results with graph-based word sense induction. Computational Linguistics 39(3), 709–754 (2013)
6. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD. pp. 133–142 (2002)
7. Lappas, T., Arai, B., Platakis, M., Kotsakos, D., Gunopulos, D.: On burstiness-aware search for document sequences. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 477–486. ACM (2009)
8. Lee, P., Lakshmanan, L.V., Milios, E.E.: Incremental cluster evolution tracking from highly dynamic network data. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on. pp. 3–14. IEEE (2014)
9. Lin, C., Lin, C., Li, J., Wang, D., Chen, Y., Li, T.: Generating event storylines from microblogs. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 175–184. ACM (2012)
10. Lin, J., Efron, M.: Overview of the TREC-2013 Microblog Track. In: TREC'13 (2013)
11. Lin, J., Efron, M.: Overview of the TREC-2014 Microblog Track. In: TREC'14 (2014)
12. Lv, C., Fan, F., Qiang, R., Fei, Y., Yang, J.: PKUICST at TREC 2014 Microblog Track: Feature Extraction for Effective Microblog Search and Adaptive Clustering Algorithms for TTG (2014)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119 (2013)
14. Ounis, I., Macdonald, C., Lin, J., Soboroff, I.: Overview of the TREC-2011 Microblog Track. In: TREC'11 (2012)
15. Walid, M., Wei, G., Tarek, E.: QCRI at TREC 2014:Applying the KISS principle for TTG task in the Microblog Track (2014)
16. Wang, D., Li, T., Ogihara, M.: Generating pictorial storylines via minimum-weight connected dominating set approximation in multi-view graphs. In: AAAI (2012)
17. Wang, X., Zhai, C.: Learn from web search logs to organize search results. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 87–94. ACM (2007)
18. Xu, T., McNamee, P., Oard, D.W.: Hltcoe at trec 2014: Microblog and clinical decision support (2014)
19. Zhai, C., Cohen, W.W., Lafferty, J.: Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. pp. 10–17. ACM (2003)
20. Zhang, Y.: Using bayesian priors to combine classifiers for adaptive filtering. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 345–352. ACM (2004)
21. Zhou, W., Shen, C., Li, T., Chen, S., Xie, N., Wei, J.: Generating textual storyline to improve situation awareness in disaster management. In: In Proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IRI 2014) (2014)