



Exploring hypergraph-based semi-supervised ranking for query-oriented summarization

Wei Wang^a, Sujian Li^{a,*}, Jiwei Li^a, Wenjie Li^b, Furu Wei^c

^a Key Laboratory of Computational Linguistics, Peking University, Ministry of Education, Beijing, China

^b Department of Computing, The Hong Kong Polytechnic University, Hong Kong

^c Natural Language Computing Group, Microsoft Research Asia, Beijing, China

ARTICLE INFO

Article history:

Received 9 September 2011

Received in revised form 26 February 2013

Accepted 4 March 2013

Available online 13 March 2013

Keywords:

Query-oriented summarization

Text hypergraph

Semi-supervised ranking

ABSTRACT

Traditional graph based sentence ranking algorithms such as LexRank and HITS model the documents to be summarized as a text graph where nodes represent sentences and edges represent pairwise relations. Such modeling cannot capture complex group relationship shared among multiple sentences which can be useful for sentence ranking. In this paper, we propose to take advantage of hypergraph to remedy this defect. In a text hypergraph, nodes still represent sentences, yet hyperedges are allowed to connect more than two sentences. With a text hypergraph, we are thus able to integrate both group relationship and pairwise relationship into a unified framework. Then, a hypergraph based semi-supervised sentence ranking algorithm is developed for query-oriented extractive summarization, where the influence of query is propagated to sentences through the structure of the constructed text hypergraph. When evaluated on DUC datasets, performance of our proposed approach shows improvements compared to a number of baseline systems.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Today, with the rapid growth of the World Wide Web (WWW), it has become a burden for people to read a large amount of documents from diverse sources to get the information they need. Automatic text summarization provides an effective means to access the exponentially increased collection of information. The main evaluation forums providing benchmarks for researchers working on automatic text summarization are the Document Understanding Conference (DUC) and Text Analysis Conference (TAC). Over the past years, DUC and TAC evaluations have gradually evolved from single-document to multi-document and from generic to query-oriented summarization. Query-oriented multi-document summarization aims to produce a short and concise summary for a collection of relevant documents describing a given topic or event with respect to a query that expresses the information need of users.

Although automatic text summarization has a long history that dates back to the 1960s, up to now extractive summarization that directly extracts sentences from documents to generate summaries is still the mainstream regardless of the nature and the goals of the tasks. Under this framework, sentence ranking is the most important issue for sure. In recent years, graph based sentence ranking draws considerable attention of the summarization community due to its ability to take into account the relationship between sentences and calculate sentence significance recursively through the global, rather than local, link structure of the text graph that are constructed by connecting the associated sentences together. Relationship is normally measured in terms of the cosine similarity between two sentences. Graph based sentence ranking is mainly

* Corresponding author. Tel.: +86 10 62753081 105.

E-mail address: lisujian@pku.edu.cn (S. Li).

inspired by PageRank [19] and HITS [12], which have been successfully used for ranking Web pages on the Web graph. Its advantages are also well recognized in text summarization [17,28–30].

For query-oriented summarization, as summarization is driven by a query, the query relevance must be measured in certain ways. Commonly used unsupervised graph based ranking formulates the query's effect either on sentence nodes by sentence-query relevance [17,30] or on links between sentences by query-sensitive similarity [28]. Both of these two measures need to be defined manually. Supervised graph based ranking, on the other hand, automatically learns how sentence similarity is skewed by query from the documents and the corresponding human abstracts that have already been available [18]. An obvious problem with such kind of supervised ranking is that no one can precisely measure the sentence similarity with or without consideration of the effect of the given query. A more convenient and effective way to impose the query's influence on the graph is to apply a semi-supervised learning approach. Being halfway between supervised and unsupervised learning, semi-supervised learning benefits from both labeled and unlabeled data. With semi-supervised graph based ranking, one can model the given query as the only labeled node on the graph which is independent of the other unlabeled sentence nodes. The label here indicates the query relevance. Initially, query relevance of the sentences is supposed to be unknown. Through the structure of the graph, the query relevance is then recursively propagated from the query to the sentences [6,13,14,22].

A deficiency of the conventional pairwise graph based modeling that one can observe is its inability to completely capture n -ary association among multiple sentences. For example, a group of sentences may be highly related to one another and collectively describe one aspect or subtopic of an event. Here, the group association information cannot be totally ignored in sentence ranking. However, it is not easy to directly superimpose such complex associations among multiple sentences on a conventional graph where only pairwise relationship is described. It is obvious that even richer information cannot be added into a conventional graph any more. To alleviate the problem, we propose to model sentences and their associations as a hypergraph, i.e. a generalization of the conventional graph, which is able to formulate more types of relationships including both pairwise and group ones. Two basic issues addressed in this paper are: (1) how to construct a text hypergraph for summarization and (2) how to develop a hypergraph based semi-supervised learning algorithm for sentence ranking. Then, the main contributions of our work are: (1) a more natural and appropriate text representation is investigated to characterize as much as useful associations among sentences and (2) semi-supervised ranking and hypergraph modeling are integrated into a unified summarization framework.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work on graph based summarization, the background of hypergraph modeling and its applications, and graph/hypergraph based semi-supervised learning. Section 3 explains the deficiency of graph modeling. Sections 4 and 5 then provide fundamentals of hypergraph modeling, introduce our text hypergraph construction algorithm, and detail the hypergraph based semi-supervised sentence ranking algorithm. Section 6 reports experiments and evaluations. Finally, Section 7 concludes the paper.

2. Background and related work

Extractive summarization selects sentences from documents to form summaries directly without any sort of paraphrase. For query-oriented summarization, the sentences extracted are required to not only be relevant to query but also contain the most significant information in documents.

In general, existing summarization approaches fall into two categories, i.e. feature based approaches and graph based approaches. Feature based approaches assign a significance score to each sentence and select the sentences with the highest scores into a summary. Significance is often computed by a linear combination of various query dependent and query independent features, such as similarity between sentence and query, term frequency, sentence position, centroid [21], signature term [15], and event features [31]. The difficulty of feature based approaches mainly lies in the assignment of feature weights. Manually assigned weights can hardly achieve predictable performance since there are a large amount of parameter combinations. Meanwhile, supervised learning of weights requires high quality and high quantity training data which is hard to obtain.

In recent years, inspired by link analysis based ranking algorithms, like PageRank [19] and HITS [12], which have achieved much success in Web page ranking, graph based approaches have attracted much attention in the summarization research. Compared with feature based approaches in which sentences are calculated independently, graph-based approaches take into account the associations among sentences and rely on sentence associations to calculate a significance score for each sentence. Graph based summarization approaches usually model sentences in one document or a set of documents to be summarized as a weighted text graph. Sentence significance scores are then recursively calculated according to the global structure of the entire text graph rather than individual sentences alone. For example, PageRank-like algorithms, such as LexRank [7], model documents as a stochastic graph in which the transition probability between two sentences is proportional to their similarity. Then the significance of each sentence is scored by calculating the stationary distribution of the stochastic graph. Furthermore, the topic sensitive LexRank algorithm [17] is proposed for query-oriented summarization, in which the transition probability consists of two parts: the query relevance is propagated from a sentence to its similar sentences with probability of d , and to those sentences that are similar to the query with probability of $(1 - d)$. While PageRank-like algorithms normally consider similarity or association between sentences, Zha [33] proposes to accomplish key phrase extraction and generic summarization simultaneously by modeling text as a weighted undirected bipartite graph. Then, significance scores of key phrases and sentences are generated based on mutual reinforcement of terms and sentences. Later,

Wan et al. [23,24] put Zha's work forward. They build a mixed model with sentences (or clusters) as hubs and terms as authorities. In their work, mutual reinforcement exists not only between hubs and authorities but also in hubs and authorities themselves. Wei et al. [28–30] integrate the notion of mutual reinforcement into PageRank-like algorithms. They introduce a unified mutual reinforcement chain, where reinforcement among terms, sentences and documents are considered simultaneously. Meanwhile, PNR² by Li et al. [13] adds the negative reinforcement between sentences, and MRSP by Du et al. [6] turns the historical sentences into sink points which are limited their reinforcement with other sentences.

A hypergraph is a generalization of a graph, where edges can connect any number of nodes. Hypergraph has proved to be a successful tool to represent and model complex concepts and structures in computer science. Of its various applications, hypergraph based partitioning has been widely used in many areas including relational database [9], data mining [10], VLSI design [11] and video summarization [27]. It has been concluded that hypergraph based clustering performs better than traditional graph based clustering (e.g. *K*-Means) on high dimension data [9,10]. Recently, hypergraph based document categorization has been explored as well [36]. In [36], documents to be classified are regarded as nodes on a hypergraph while each word corresponds to a hyperedge that connects all the documents containing this word. To our best knowledge, there is no previous work reported on hypergraph based text summarization yet.

Semi-supervised learning automatically exploits unlabeled data in addition to labeled data in order to improve learning performance. Among them, graph based semi-supervised learning has attracted a lot of attention recently. It is non-parametric and can easily adapt to large datasets. In graph based semi-supervised learning, a graph is defined with nodes representing labeled and unlabeled examples in the dataset and edge weights reflecting the similarity between examples. The goal of graph based semi-supervised learning is to estimate a function on the graph, which can then be used to predict node labels. Based on the prior assumption [34] that nearby points and the points on the same manifold are likely to have the same label (or similar scores), the function to be estimated is required to be consistent with labeled data and meanwhile locally smooth on the graph. These two constraints (called fitting constraint and smoothness constraint, respectively) are usually formulated by a regularization function that consists of a fitness regularizer and a smoothness regularizer. Up to now, the smoothness regularizer is the focus of research and different smoothness regularizers have been explored, such as combinatorial graph Laplacian in the Gaussian random fields and harmonic function method [36], normalized graph Laplacian in the local and global consistency method [34], and local linear regularization in the linear neighborhood propagation (LNP) method [26].

The success of graph based semi-supervised learning has prompted researchers to further study semi-supervised learning on hypergraph. For example, Corduneanu and Jaakkola [4] address a more general semi-supervised classification problem where a label distribution instead of a fixed label is learned for each node on a hypergraph. They propose a regularization function based on the Kullback–Leibler (KL) divergences among distributions on a hyperedge. Zhou et al. [35] also propose the hypergraph Laplacian which is deduced from a hypergraph based smoothness function similar to the one they use for graph in the local and global consistency method. Their hypergraph Laplacian has proved to be quite effective and been successfully applied in the tasks like automatic image annotation [25] and image segmentation [5]. Zhou et al.'s work is fundamental to the hypergraph based semi-supervised sentence ranking algorithm introduced in Section 6. Our focus in this work, however, is semi-supervised ranking, other than classification.

3. Problem statement

Pairwise relationships among the objects to be ranked are generally assumed in the traditional graph based ranking algorithms, where each node in the graph stands for an object and two nodes are connected by an edge if there exist some kind of relationship between them. Under this assumption, it is natural to use a graph to represent a set of objects endowed with pairwise relationships. The graph can be directed or undirected, depending on whether the relationships among the objects are symmetric or asymmetric. The text graph constructed upon symmetric sentence similarity is a typical example of undirected graphs, whilst a well-known instance of directed graphs is the WWW, where pages are connected by directed hyperlinks.

However, in many real world applications, the relationships among the objects are more complicated than simple pairwise ones. For example, in interpersonal networks, there exist not only pairwise relationships like “friend” and “lover”, but also group relationships like “family” and “organization”, which normally involve more than two participants. It is important and challenging to simultaneously exhibit various types of relations, not limited to only one type of pairwise relations. One way of modeling such rich information is to represent the objects and their relations in a hypergraph instead. At the same time, hypergraph models have achieved much success in classification and clustering for complex relational data.

Similarly, we have an analogous situation in graph based summarization. Most previous graph based summarization approaches use conventional text graph to represent one document or a set of documents to be summarized, in which each node stands for a sentence and the edge weight is defined as the similarity between two sentences. The graph simply built on pairwise sentence similarity cannot fully capture the information carried in the documents. For example, a given topic or event usually covers a few aspects or **subtopics**¹ [32]. Each subtopic is composed of a set of sentences each of which plays the same role in expressing the subtopic. Fig. 1 shows a hypergraph example where five hyperedges (i.e. e_1 , e_2 , etc.) represent five subtopics for a given event. However, such subtopic information is difficult to be represented in the conventional pairwise

¹ The notion of subtopic in our paper is the same as that of topic in some referenced papers.

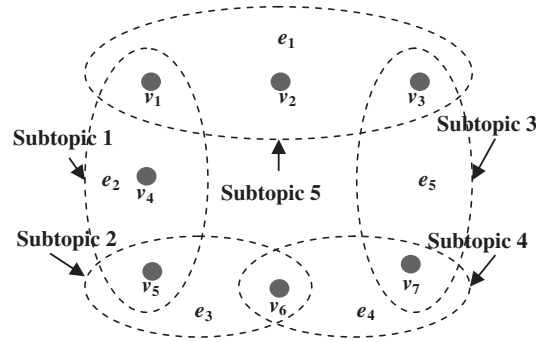


Fig. 1. Hypergraph representation of sentences describing one event.

similarity graph. Wan and Yang [24] has made use of the subtopic information in a two-level graph and illustrated that the addition of subtopics can effectively improve the summary quality. In this paper, we mainly explore how to incorporate the subtopic information into a hypergraph. We believe that hypergraph can provide a more natural and appropriate way to model non-pairwise relationships. More precisely, the group relationships can be easily incorporated by adding a hyperedge to connect the sentences in the same group. Meanwhile, a hypergraph still allows the connections, which are held between the two sentences with pairwise relations, to be retained. Through this text representation, both the “local” pairwise similarity information and the “global” group commonality information such as subtopic can be preserved.

4. Text hypergraph modeling

4.1. Fundamentals of hypergraph

A **hypergraph** is a generalization of the conventional graph. Hyperedges are arbitrary subsets of the node set, i.e. the hyperedges can connect two or more nodes on a hypergraph. Formally, a hypergraph can be defined as $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes and $E = \{e_1, e_2, \dots, e_m\}$ is a set of hyperedges, where $e_i \subseteq V$ for $i = 1, 2, \dots, m$. Clearly, the hypergraph is degenerated into a standard graph when $|e_i| = 2$ for all $i = 1, 2, \dots, m$. The sizes of V and E are denoted by $|V|$ and $|E|$, respectively.

A **weighted hypergraph** $G = (V, E, W)$ is a hypergraph that has a positive weight $w(e)$ associated with each hyperedge $e \in E$. For a node $v \in V$, its degree is defined by

$$d(v) = \sum_{\{e \in E | v \in e\}} w(e) \tag{1}$$

And the degree of $e \in E$ is defined to be

$$\delta(e) = |e| \tag{2}$$

A hyperedge e is said to be incident with a node v if $v \in e$. It is worth noting that there is one to one correspondence between a hypergraph and a 0–1 matrix. Given a hypergraph, its matrix representation H , called the **incidence matrix**, is defined as

$$h(v, e) = \begin{cases} 0, & \text{if } v \notin e \\ 1, & \text{if } v \in e \end{cases} \tag{3}$$

For example, the hypergraph illustrated in Fig. 1 can be represented using the following incidence matrix:

	e_1	e_2	e_3	e_4	e_5
v_1	1	1	0	0	0
v_2	1	0	0	0	0
v_3	1	0	0	0	1
v_4	0	1	0	0	0
v_5	0	1	1	0	0
v_6	0	0	1	1	0
v_7	0	0	0	1	1

With the incidence matrix, the degrees of the node v and the edge e can be simply denoted as

$$d(v) = \sum_{e \in E} w(e)h(v, e) \tag{4}$$

and

$$\delta(e) = \sum_{v \in V} h(v, e) \quad (5)$$

Let D_v and D_e denote the diagonal matrices containing the node and the hyperedge degrees respectively, and W denote the diagonal matrix containing the hyperedge weights. Then the **adjacency matrix** A of G is defined as $A = HWH^T - D_v$. Notice that when we mention a hypergraph in the rest of the paper, we refer to an undirected text hypergraph built upon similarity measures unless otherwise specified.

4.2. Construction of text hypergraph

Unlike in the applications such as citation analysis where the construction of one hypergraph based on the relationships among authors and articles is straightforward, the construction of text hypergraph is not direct but requires an additional process to discover group relationships. In this paper, we mainly explore how to combine the subtopic information in a hypergraph, though there are many different ways to define such group relationships among the sentences. Same as Wan's work, we adopt the clustering algorithms in the representation of the subtopics.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a density based clustering algorithm proposed by Easter et al. [8]. The key idea of Easter's DBSCAN is that for each node in a cluster (except for those border nodes), the neighborhood of a given radius should contain at least a minimum number of nodes. Or rather, the density around the node should exceed a given threshold. The reason we choose DBSCAN for clustering is twofold. First, DBSCAN does not need a predefined cluster number as the input parameter. It can automatically determine the number of clusters during the clustering process. Second, unlike other clustering algorithms which endow each node with a cluster id, DBSCAN is capable of filtering the noise nodes, i.e. the nodes not belonging to any cluster. Using DBSCAN, we need to assign appropriate values to two basic parameters, namely Eps and MinPts, which defines the search radius from each node and the minimum number of the nodes that should be contained in the neighborhood, respectively. According to the directly density-reachable property of DBSCAN, the value of MinPts means that a cluster is composed of at least MinPts nodes. It seems unreasonable that too few nodes comprise one cluster and MinPts is usually empirically set as 4. To automatically tune the parameter Eps in DBSCAN, we apply a modified DBSCAN algorithm to cluster the sentences.

Formally, a document set $D = \{s_1, s_2, \dots, s_n\}$ containing n sentences and an associated query s_0 , each sentence s_i is represented by a weighted term vector under the VSM (vector space model) scheme. The weight of a term is calculated using the inverse sentence frequency (ISF) [1] with the intuition that the terms contained in a few sentences should be more important than those appearing in many sentences. Let sf_t denote the number of the sentences that contain the term t , then ISF of t is calculated by $isf_t = \log((n+1)/sf_t)$. The distance between two sentences s_i and s_j ($0 \leq i, j \leq n$), which is fundamental to DBSCAN, is then defined as

$$dist(s_i, s_j) = 1 - cosine(s_i, s_j) \quad (6)$$

where $cosine(s_i, s_j)$ is the cosine similarity between s_i and s_j , and $cosine(s_i, s_j) = s_i \cdot s_j / (|s_i| |s_j|)$. Since the $cosine(s_i, s_j)$ value ranges from 0 to 1, the value of $dist(s_i, s_j)$ is between 0 and 1 as well. According to the weighted terms, the whole documents are separated into several clusters as well as a noise set, each of which is composed of a set of sentences. It is ideal that short sentences with little information and sentences away from the topic are seen as noise.

The clustering results of DBSCAN, generally, are quite sensitive to the settings of parameter Eps. On the one hand, given a very large Eps value, say 0.99, almost all the sentences will be grouped into one cluster. On the other hand, if we choose a small value as the search radius, then almost all the sentences will be detected as noise. Since the scope of subtopics varies from one to another, it is hard to define a fixed Eps value as search radius. A feasible solution is to start clustering with a larger Eps value, say 0.8. Each time when a cluster is formed, its clustering result is examined to see if it is "reasonable". The criterion of judging whether the clustering result is reasonable is that no cluster is allowed to include a large number of sentences (e.g. one third of all the sentences) in a document set. If this constraint is not satisfied, the sentences will be re-clustered with an automatically adjusted smaller Eps value until a reasonable clustering result is obtained. Then the Eps value can be automatically tuned during the re-clustering process with the complexity of $O(tn \log n)$ where t means the iteration times and n means the number of sentences. We call this derived clustering algorithm as the modified DBSCAN algorithm. It is depicted in Fig. 2.

The modified DBSCAN algorithm assures that there is no dominant cluster, i.e. the cluster including the majority of the sentences. The clustering result looks rather reasonable when we look through them carefully. For example, for the DUC topic D0601A (i.e. the topic of native American Reservation system), out of a total of 1132 sentences (including the query), 733 are identified as noise. This is rather reasonable since many sentences are either too short or do not carry enough information. The rest 399 sentences are grouped into 25 clusters. The size of the largest cluster is 162, while the sizes of the other 24 clusters range from 4 to 25. We can see that these 25 clusters have described the different subtopics of Native American Indians. For instance, the largest cluster (Cluster 3) is mainly about the common problems existed in the Indian communities, such as unemployment, crime and drug abuse, while Cluster 1 focuses on the progress that has been made in an Indian town called Pine Ridge, and Cluster 8 talks about the government budget for American Indian reservations. These clusters are composed of the subtopics (hyperedges) illustrated in Fig. 1.

Algorithm 1: *ModifiedDBSCAN(D, query)*

Input: The document set $D = \{s_1, s_2, \dots, s_n\}$ and $query=s_0$

Output: The cluster set $C = \{c_1, c_2, \dots, c_k\}$

MODIFIED – DBSCAN

Let $D' = D \cup \{s_0\}$;

Let MinPts = 4, Eps = 0.80, ContinueIteration = true;

WHILE Continue Iteration

$C = DBSCAN(D', Eps, MinPts)$;

Let maxcluster = $\max(|c_j| \mid 1 \leq j \leq k)$;

IF maxcluster > $|D'| * 0.333$

Eps = Eps – 0.01;

ELSE

ContinueIteration = false;

ENDIF

ENDWHILE

Return C ;

END

Fig. 2. The modified DBSCAN algorithm.

Once the clusters are obtained, we construct the text hypergraph as follows: (1) if the cosine similarity between the two sentence s_i and s_j is positive, then we add a hyperedge to connect them with the weight $w(e)$ as $\text{cosine}(s_i, s_j)$ and (2) C is composed of all the clusters (i.e. c_1, c_2, \dots, c_k , etc.) generated by the modified DBSCAN. For the sentences $s_{t_1}, s_{t_2}, \dots, s_{t_k}$ ($0 \leq t_i \leq n, 1 \leq i \leq k$) which are in the same cluster c_t , we add a hyperedge e to connect them with the weight $w(e)$ defined as the cosine similarity between the cluster c_t and the whole document, i.e. $\text{cosine}(c_t, D)$. In addition, the elements in incident matrix H , edge degree matrix D_e and vertex degree matrix D_v , are set their corresponding values. We can see that b is a parameter balancing the relative importance of cluster hyperedges compared with pairwise hyperedges. The **ConstructHypergraph** algorithm has the complexity of $O(n \log n)$ as in Fig. 3.

5. Hypergraph based semi-supervised learning for sentence ranking

It has been well acknowledged that sentence ranking is the issue of most concern under the extractive summarization framework. Sentence ranking has long been addressed in an unsupervised manner. Facilitated by the advances in machine

Algorithm 2: *ConstructHypergraph(D, query, b)*

Input: The document set $D = \{s_1, s_2, \dots, s_n\}$, $query=s_0$ and the parameter b

Output: The hypergraph incident matrix H , the diagonal hyperedge weight matrix W , hypergraph edge degree matrix D_e and hypergraph vertex degree matrix D_v

ConstructHypergraph

Let $D' = D \cup \{s_0\}$;

$C = ModifiedDBSCAN(D')$;

FOR each sentence s_i in D'

FOR each sentence $s_j (j > i)$ in D'

IF $\text{cosine}(s_i, s_j) > 0$,

Add a hyperedge e ;

Let $w(e) = \text{cosine}(s_i, s_j)$;

Let $h(s_i, e) = 1, h(s_j, e) = 1$

Let $h(s_k, e) = 0$, for $0 \leq k \leq n, k \neq i, j$;

ENDIF

ENDFOR

FOR each cluster c_k in C

Add a hyperedge e ;

Let $w(e) = b * \text{cosine}(c_k, D')$;

Let $h(s_i, e) = 1$, if $s_i \in c_k$, and $h(s_i, e) = 0$, if $s_i \notin c_k$;

ENDFOR

Let $D_v(s_i) = \sum_{e \in E} w(e)h(s_i, e)$, $D_e(e) = \sum_{s' \in D'} h(s', e)$;

Return H, W, D_v and D_e ;

END

Fig. 3. The algorithm of constructing text hypergraph.

learning, supervised and semi-supervised approaches have been investigated in recent past to learn the sentence significance for ranking [3,33]. The bottleneck of learning based sentence ranking is the acquisition of training data. Manual annotation of training data is a complex and arduous task both time-consuming and costly. In this paper, we explore how to take advantage of the existing labeled sentences to develop the hypergraph based semi-supervised sentence ranking for labeling those unlabeled sentences in query-oriented summarization.

Given the hypergraph representation of a document set, one way to make convenient and better use of the information carried on the hypergraph is to treat sentence ranking as a semi-supervised learning process, in which the query is regarded as the only labeled node, whilst the sentences in documents as unlabeled nodes. In fact, the only information we have at the very beginning about how to select the most significant sentences to generate one summary is the query alone. Hence, initially the query is assigned with a positive significance score (e.g. 1 in Algorithm 3) and the significance scores of the sentences in the documents are assigned zero. Then sentence scores are learned step by step from the query through the structure of the hypergraph. The key idea behind hypergraph based semi-supervised ranking is that the nodes which have many incident hyperedges in common should be assigned with very similar scores. This just fits well in the context of summarization. In particular, we have the following two assumptions (or to say constraints):

1. The sentences with higher cosine similarity should have significance scores that are more similar to each other.
2. The sentences in the same cluster that talk about the same topic should have similar scores.

Given a weighted hypergraph $G = (V, E, W)$, which represents the documents to be summarized, and a scoring function f over V , which assigns the sentence v the score $f(v)$. We may think of f as a vector in Euclid space $R^{|V|}$. Then, the functional $\Omega(f)$ is defined to formalize the above-mentioned two assumptions.

$$\Omega(f) = \frac{1}{2} \sum_{e \in E} \frac{1}{\delta(e)} \sum_{\{u,v\} \subseteq e} w(e) \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2 \quad (7)$$

where $\{u, v\} \subseteq e$ means that two edges u, v are included in the same edge e , and $\delta(e)$ and $w(e)$ respectively denote the degree and weight of edge e . The function $\Omega(f)$ sums the changes of the scoring function f over the hyperedges on the hypergraph. On the one hand, a good scoring function should make $\Omega(f)$ as small as possible, i.e. the sentences which share higher cosine similarity or exist in the same cluster should have scores that are more similar to each other. On the other hand, a good scoring function should be consistent with the given initial score vector. As mentioned previously, the initial score of the query is

Algorithm 3: *HyperSum*($D, query, a, b$)

Input: The document set $D = \{s_1, s_2, \dots, s_n\}$, $query = s_0$, parameters a and b

Output: The summary of the document set $Sum = \{s_{i_1}, s_{i_2}, \dots, s_{i_l}\}$

HyperSum

Let $H, W, D_v, D_e = \text{ConstructHypergraph}(D, s_0, b)$;

Let $\theta = D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}$;

Set $Sum = \emptyset$, and the summary length $l = 0$;

Initialize the score vector y as $y_0 = 1$, and $y_i = 0$, for $1 \leq i \leq n$;

Let $f^0 = y$;

REPEAT

$f^{k+1} = a\theta f^k + (1-a)y$;

$\nabla \leftarrow \max_i |f_i^{k+1} - f_i^k|$, for $0 \leq i \leq n$;

$k \leftarrow k + 1$;

UNTIL $\nabla < \zeta'$

Sort the sentences $\{s_1, s_2, \dots, s_n\}$ by their significance scores, Let

$T = \{s_{t_1}, s_{t_2}, \dots, s_{t_n}\}$ such that $f_{t_1}^* \geq f_{t_2}^* \geq \dots \geq f_{t_n}^*$;

FOR $i \leftarrow 1$ to n do

threshold $\leftarrow \max(\text{cosine}(s_{t_i}, s) | s \in Sum)$;

IF threshold ≤ 0.7 do

$l \leftarrow l + \text{length}(s_{t_i})$;

IF ($l < \text{limit}$) $Sum \leftarrow Sum \cup \{s_{t_i}\}$; else BREAK; ENDIF

ENDIF

ENDFOR

Return Sum ;

END

Fig. 4. HyperSum: hypergraph based semi supervised ranking summarization algorithm.

-
- (1) With economic opportunities on reservations lagging behind those available in big cities, and with the unemployment rate among Native Americans at three times the national average, thousands of poor, often unskilled Native Americans are rushing off their reservations.
 - (2) Federal programs distributed to American Indians based on census data include the Native American Employment and Training Programs, grants to local education agencies for Indian education, and family violence prevention and services.
 - (3) And the state also supports the National American Indian Association here.
 - (4) Native Americans living on reservations that maintain 50 percent or more unemployment are exempt from the national five-year family limit on welfare benefits.
 - (5) Also, many Native Americans travel between the reservation and urban areas.
 - (6) Smith and thousands like her are seeking help for their substance abuse at the American Indian Community House, the largest of a handful of Native American cultural institutions in the New York area.
 - (7) But sovereignty clearly means recognizing Indians as not only Native Americans but also as special Americans.
 - (8) Neah Bay, like many other Native American communities, is struggling with unemployment rates often exceeding 50 percent, deep poverty and the problems of crime, drug and alcohol abuse and domestic violence.
 - (9) As crime rates fall nationwide, they are rising in American Indian communities, especially among the 43 percent of Indians under age 20.
 - (10) An estimated 50 percent of American Indians are unemployed, and at Pine Ridge the problem is even more chronic -- 73 percent of the people do not have jobs.
-

Fig. 5. The summary generated for Topic D0601A.

assumed to be 1 and the initial scores of the other sentences 0. Let y denote the initial score vector, then the significance scores of the sentences are learned recursively by solving the following optimization problem:

$$\operatorname{argmin}_{f \in \mathbb{R}^{|V|}} \{ \Omega(f) + u \|f - y\|^2 \} \quad (8)$$

where $u > 0$ is the parameter specifying the tradeoff between the two competitive terms. Zhou has proved that this optimization problem has a closed form solution [35].

$$f^* = (1 - a)(I - a\Theta)^{-1}y \quad (9)$$

where $a = 1/(1 + u)$ and $\Theta = D_v^{-1/2}HWD_e^{-1}H^T D_v^{-1/2}$. The above equation can then be re-formulated as:

$$f^* = a\Theta f^* + (1 - a)y \quad (10)$$

Eq. (10) can be explained from another perspective. First, we assign a positive significance score to the query node and zero scores to the remaining nodes. All the nodes then spread their significance scores out to their nearby neighbors via the hypergraph. The weights of the transitions between any two nodes are defined by Θ . a is a parameter that specifies the proportion of how much a node should learn from its neighbors, and how much it should learn from the specified initial scores of the labeled nodes. The propagation process is repeated until a global stable state is achieved. Then all the nodes obtain their final significance scores. To compute f^* , we apply an iterative approach, which assigns f^* with an initial value and then calculate the new value of f^* by Eq. (10) until the convergence is reached.

Once the sentence significance scores are ready, we rank sentences in descending order of the calculated scores and the sorted sentences $T = \{s_{t_1}, s_{t_2}, \dots, s_{t_n}\}$ from which the highest ranked sentences are picked into a summary which is denoted by a sentence set SUM . To avoid the redundancy in the generated summary, each time when the top ranked sentence s_{t_i} is examined as a candidate summary sentence, the cosine similarity between it and each of the previously selected summary sentences $s \in SUM$ is calculated. If the similarity $\cosine(s_{t_i}, s)$ exceeds a pre-defined threshold (e.g. 0.7), the candidate sentence is discarded. This procedure is repeated until the summary length l reaches the limit of words (e.g. 250 words). We call this query-oriented summarization approach based on the hypergraph representation and semi-supervised learning paradigm *HyperSum*. The algorithm of *HyperSum* is detailed in Fig. 4.

We still take topic D0601A (i.e. the topic of native American Reservation system) for example, Fig. 5 lists the generated summary which is composed of 10 sentences. From the figure, we can see that Sentences (1), (5) and (8) are from Cluster 3 talking about the common problems existed in the Indian communities, Sentences (2) and (4) are from Cluster 8 involving the government actions for the reservations, and Sentence (10) is from Cluster 1 about an Indian town called Pine Ridge. Normally, important sentences are preferred to be extracted from important clusters.

6. Experiment and evaluation

6.1. Experiment set-up

The query-oriented multi-document summarization task defined in DUC evaluations requires generating a concise and well organized summary for a set of topic related documents according to a query which describes the users' information need. The query usually consists of a title and one or more narrative/question sentences.

Table 1
Experiment data.

DUC	2005	2006	2007
Collection #	50	50	45
Avg Doc # per docset	32	25	25
Avg Sen # per coll.	914	705	499

Our experiments are conducted on the DUC 2005, 2006 and 2007 data sets, which respectively contain 50, 50 and 45 document sets (docsets). The DUC 2005 data set has 25–50 documents (averaged 32 documents) per docset while DUC 2006 and 2007 data sets contain 25 documents per docset, as illustrated in Table 1. The length of one system-generated summary is strictly limited to 250 words. Stop-words in both documents and queries are removed using a stop-word list of 598 words. The remaining words are stemmed by Porter Stemmer² and considered as terms. As for the evaluation metric, it is difficult to come up with a universally accepted method to measure the quality of system-generated summaries. In this work, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics [16], especially ROUGE-2 and ROUGE-SU4³ and their corresponding 95% confidential intervals, which have been officially adopted by the DUC, is used to automatically evaluate the content of system-generated summaries.

In this section, we present the performance evaluation of five sets of experiments which are conducted: (1) to examine the influence of the two parameters in *HyperSum*, i.e. the learning factor a and the cluster importance factor b , on summarization performance; (2) to analyze the performance of different clustering algorithms in addition to the modified DBSCAN algorithm in hypergraph construction; (3) to analyze the performance using different similarity metrics in hypergraph construction; (4) to verify the advantage of the text hypergraph representation by comparing the performance of *HyperSum* with the performance of the conventional graph based ranking algorithms like topic sensitive PageRank, HITS and manifold which have previously been adopted in query-oriented summarization; and (5) to compare *HyperSum* with the state-of-the-art DUC participating systems.

6.2. Parameter tuning

The goal to conduct the following experiments is to find the appropriate values for the two parameters in *HyperSum*, i.e. the learning factor a ($0 < a < 1$), and the cluster importance factor b ($b > 0$). The combination of the two factors makes it hard to find a global optimized solution. So we apply a gradient search strategy. At first, the cluster importance factor b is set to a value, e.g. $b = 1.0$. Then the performance using different values of a ranging from 0.1 to 1 is evaluated. After that, we fix a with the value which has achieved the best performance, and conduct experiments to find an appropriate value for b in the range from 0.0 to 2.0.

6.2.1. The learning factor a

First of all, the cluster importance factor b is fixed to 1, i.e. the cluster hyperedge is as important as the pairwise cosine similarity hyperedge. Figs. 6 and 7 present the ROUGE-2 and ROUGE-SU4 evaluation results of *HyperSum*, with regard to different values of a on the DUC 2006 data set. We can see that the performance of *HyperSum* gets better as a increases from 0.1 to 1. It reaches the peak at around 0.97–0.98 and drops afterwards. As mentioned in Section 4, a can be deemed as a factor that specifies the proportion of how much a sentence should learn from its neighbors and how much it should learn from the specified initial score vector. The experimental results suggest that a sentence learns its significance score mainly from its neighbor sentences. However, this does not mean that the initial score vector y is not important. As we can see, after reaching its peak, as a approaches to 1, the performance gets worse. This phenomenon can be explained as follows: for query-oriented summarization, an ideal summary is required to contain the salient sentences that not only represent the main themes of the documents but are also relevant to the query. As for *HyperSum*, it is the initial score vector that provides the guidance of the query for sentence scoring, with the intensity of this guidance specified by $(1 - a)$. As a gets nearer to 1, the query's influence gets weaker, resulting in the degradation of system performance.

6.2.2. The cluster importance factor b

Next, we fix the learning factor a at 0.98. Figs. 8 and 9 illustrate the performance of *HyperSum* on the DUC 2006 data set with respect to the different values of b ranging from 0.0 to 2.0. When $b = 0$, it means that the weight of the cluster hyperedge is 0 and thus only the pairwise cosine similarity information is considered for the calculation of sentence significance scores. As we can see, when the cluster information is considered, i.e. $b > 0$, the performance is improved in most cases. This justifies the use of text hypergraph, which incorporates the group information and the pairwise information, to represent the

² The porter stemmer is downloaded from <http://tartarus.org/simmartin/PorterStemmer/>. Here we do not further involve the meaning ambiguity of the stemmed words, since the disambiguation cost is high and may not bring much performance improvement.

³ Jackknife scoring for ROUGE is used in order to compare with the human summaries.

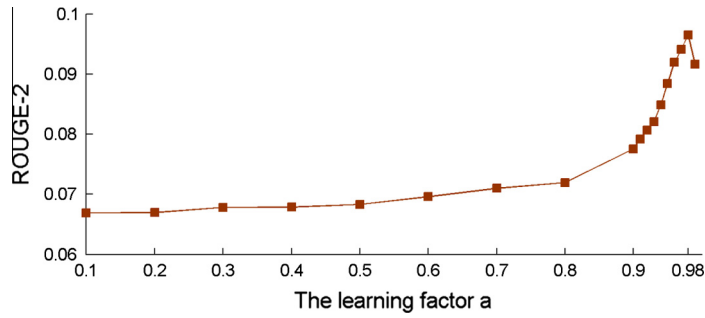


Fig. 6. ROUGE-2 with $a = [0.1, 0.99]$.

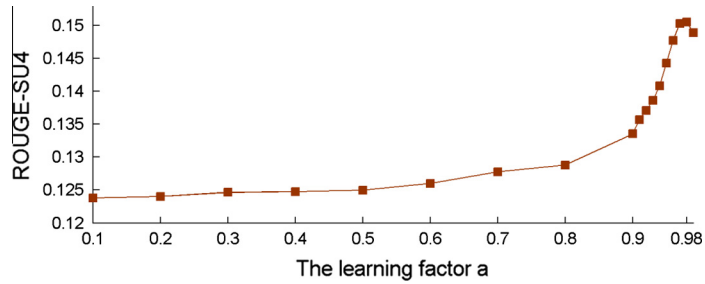


Fig. 7. ROUGE-SU4 with $a = [0.1, 0.99]$.

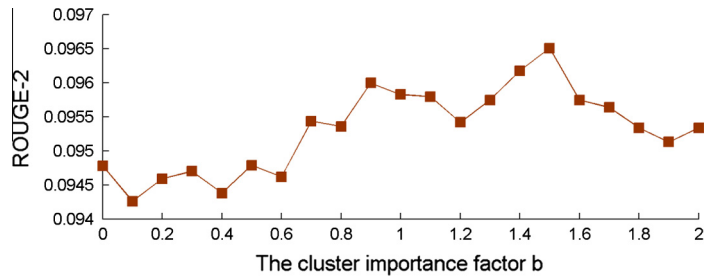


Fig. 8. ROUGE-2 with $b = [0, 2]$.

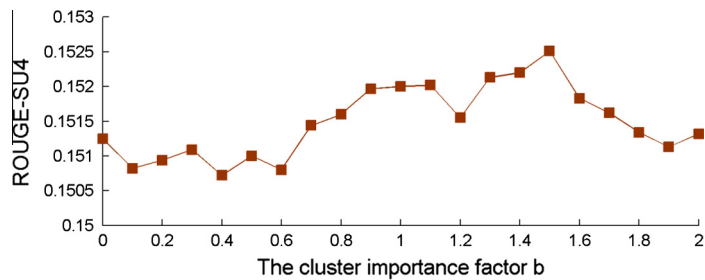


Fig. 9. ROUGE-SU4 with $b = [0, 2]$.

documents. Besides, it is observed that the best performance is achieved when b is set to 1.5. This result suggests that cluster hyperedges should be assigned higher weights than pairwise cosine similarity hyperedges.

6.3. Comparison of clustering algorithms

In this set of experiments, four additional clustering algorithms are explored for hypergraph construction and they are compared against the proposed modified DBSCAN algorithm. The four selected algorithms include K -means clustering, latent

Dirichlet allocation (LDA), agglomerative clustering, and divisive clustering, which perform the clustering and organize the clusters in different ways.

6.3.1. *K-means clustering*

It is a partition based clustering algorithm. It first randomly picks k nodes as cluster centroids, assigns each node to the nearest cluster and then re-computes the cluster centroids. The above steps are repeated until the cluster centroids do not change any more.

6.3.2. *Latent Dirichlet allocation*

LDA is a hierarchical Bayesian model which models the word level, document level and corpus level into a uniform framework [2]. In our sentence extractive summarization task, we see each sentence as one document and LDA can cluster each sentence with a probability distribution over subtopics. The subtopic with the highest probability is taken as the cluster which the sentence belongs to.

6.3.3. *Divisive clustering*

It is a top-down hierarchical clustering algorithm, which starts with an all-inclusive cluster. During each step, the largest cluster is bisected so that the resulting 2-way clustering solution optimizes under a particular clustering criterion.

6.3.4. *Agglomerative clustering*

It is bottom-up hierarchical clustering algorithm which starts with each node as a single cluster. At each step, two clusters which are the most similar are merged until the clusters are too far apart to be merged.

Notice that both *K-means clustering* and LDA require a pre-defined cluster number k . However, given a document set, it is not easy to predict the cluster number in advance. One commonly used method is setting the cluster number as \sqrt{n} [24]. Here we also tested different cluster numbers changing from 5 to 50 with intervals of 5 when the parameters a and b are 0.98 and 1.5 respectively, and the cluster number with which the summary gets the highest ROUGE-2 score is assumed as the optimum value of the corresponding document set. These two methods of setting the cluster numbers are combined with *K-means clustering* and LDA which are denoted as *Kmeans-sqrt*, *Kmeans-best*, *LDA-sqrt* and *LDA-best* respectively in Figs. 10 and 11. Divisive clustering and agglomerative clustering do not require the specification of cluster number and are implemented using the free software Cluto.⁴ The cluster similarity used in these three algorithms is calculated by the cosine similarity between the two clusters, i.e. $\text{cosine}(c_i, c_j)$.

Figs. 10 and 11 plot the ROUGE-2 and ROUGE-SU4 results of the five clustering algorithms, given that the parameter b changes from 0 to 2 (the value of learning factor a is fixed as 0.98). From the two figures, we can see that modified DBSCAN outperforms the other four algorithms. Moreover, in most cases, by using the clustering results of the algorithms other than modified DBSCAN, the system's performance degrades or almost keeps unchanged compared with that when the cluster information is not considered, i.e. $b = 0$. This conforms to our expectation, because except for modified DBSCAN, all the other algorithms by nature are not capable of detecting noise, i.e. they may form the clusters containing many noise sentences. Furthermore, in DUC data sets, there are many noise sentences which are not suitable to be selected as the summary sentences. Since members of the same cluster tend to have similar scores, these noise sentences are likely to be assigned with a high significance score and then picked up into the summary incorrectly.

6.4. Comparison of similarity metrics

In our work, the foundation of constructing a hypergraph is the relationship between sentences, which is implemented through similarity computation. In this subsection, we aim to verify whether the hypergraph based semi-supervised ranking can be adapted to other similarity metrics. Here, we experiment on DUC2006 with two other commonly used similarity metrics, namely the *Jaccard* similarity and the *Euclidean* similarity respectively. The formula to calculate these two similarities are listed as follows:

$$\text{JaccardSim}(s_i, s_j) = \frac{\sum_{t_k \in (s_i \cap s_j)} \min(\text{freq}_{ki}, \text{freq}_{kj}) * isf_k}{\sum_{t_l \in (s_i \cup s_j)} \max(\text{freq}_{li}, \text{freq}_{lj}) * isf_l} \quad (11)$$

$$\text{EuclidSim}(s_i, s_j) = \begin{cases} 0, & \text{if } s_i \cap s_j = \emptyset \\ 1/|s_i - s_j|, & \text{else} \end{cases} \quad (12)$$

where s_i and s_j are two sentences to be measured which respectively contain a set of words, freq_{ki} denotes the frequency of term t_k occurring in sentence s_i and $|s_i - s_j|$ represents the Euclidean distance of s_i and s_j .

The *Jaccard* similarity and the *Euclidean* similarity are respectively applied in the construction of the text hypergraph which is applied in *HyperSum*. Then, we still adopt the gradient search strategy to find the appropriate parameters: Fixing b to the value of 1.0, the performance using different values of a ranging from 0.1 to 1 is evaluated, and then with the fixed

⁴ <http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download>.

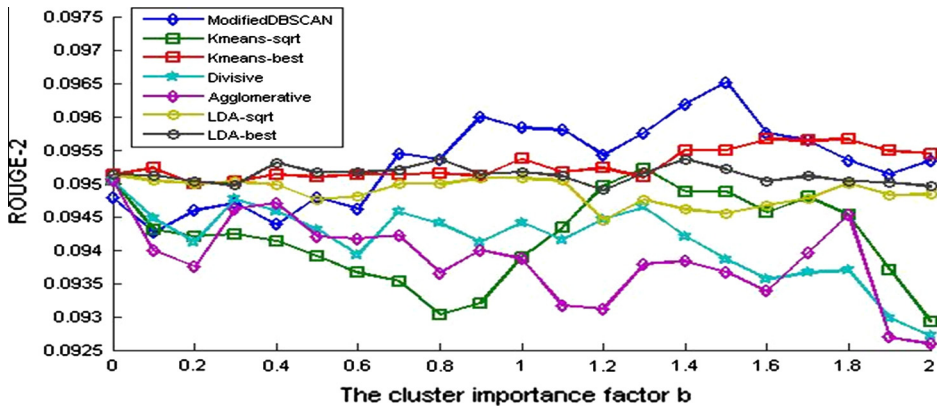


Fig. 10. ROUGE-2 with different clustering algorithms.

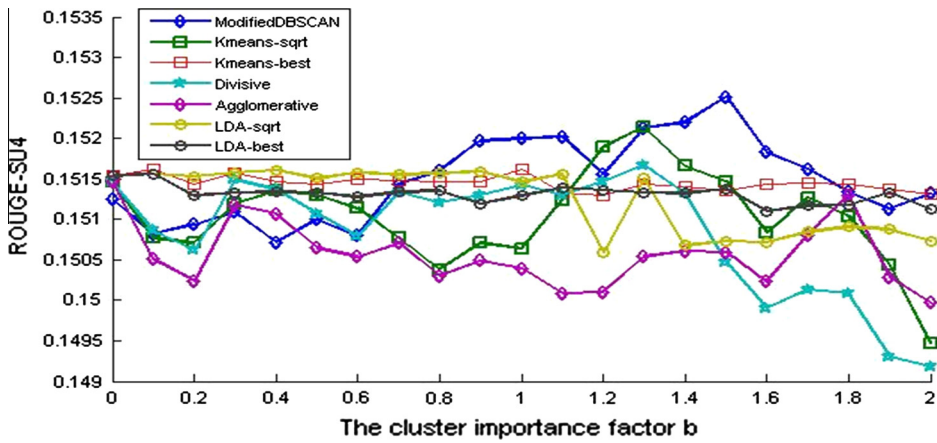


Fig. 11. ROUGE-SU4 with different clustering algorithms.

value of a , an appropriate value for b in the range from 0.0 to 2.0 is searched. Through observing the experimental results, we find that the performance curves using the *Jaccard* and *Euclidean* metrics exhibit a similar form with those using the *Cosine* metric. The optimum values of a and b are also set as 0.98 and 1.5 respectively. Table 2 shows the peak performance using three similarity metrics, and we can see that the *cosine* metric exhibits obvious advantage of measuring sentence similarity compared with the other two metrics.

6.5. Graph model vs. hypergraph model

To verify the effectiveness of the proposed hypergraph based sentence ranking algorithm, we compare *HyperSum* with two conventional graph based sentence ranking algorithms: one is developed based on topic sensitive PageRank, called query sensitive LexRank [16], and the other one is developed based on Kleinberg’s HITS [12].

Query sensitive LexRank represents the sentences in a document set as a text graph. Each sentence node on the graph is viewed as a state in a Markov chain. From the current sentence, a transition is made to the sentences that are similar to the query q with the probability of d , and a transition is made to the sentences that are lexically similar to the current sentence with probability of $(1 - d)$. The transition probability between the two sentences, s_i and s_j , is defined as:

Table 2
Comparison of similarity metrics.

	ROUGE-2	ROUGE-SU4
Cosine	0.09569 (0.08722–0.10404)	0.15182 (0.14424–0.15899)
Jaccard (1, 1.5)	0.08235 (0.07292–0.09240)	0.13772 (0.12949–0.14644)
Euclidean (1, 1.5)	0.07475 (0.06663–0.08362)	0.13294 (0.12569–0.14058)

$$p(s_i \rightarrow s_j) = (1 - d) * \frac{\text{sim}(s_i, s_j)}{\sum_k \text{sim}(s, s_k)} + d * \frac{\text{sim}(s_j, q)}{\sum_k \text{sim}(s_k, q)} \quad (13)$$

The significance score of each sentence is then obtained by calculating the stationary distribution of the Markov chain. In our implementation of the query sensitive LexRank algorithm, d is set to 0.15 as in classical PageRank.

When implementing HITS for query-oriented summarization, we simply select top 10% of the high query relevant sentences to construct the text graph according to our experience. Query relevance is defined as the cosine similarity between the sentence and the query. Sentences are regarded as hub nodes and terms appearing in the sentence are regarded as authority nodes. By applying the mutual reinforcement principle as used in HITS, the ranks of sentences and terms are calculated simultaneously and iteratively.

$$V_s = A \cdot V_t, \quad V_t = A^T \cdot V_s \quad (14)$$

where V_s and V_t are the sentence score vector and term score vector respectively, A is the affinity matrix for sentences and terms, i.e. if sentence s_i contain term t_j , then $A_{ij} = \text{isf}_{t_j} * \text{tf}_{ij}$ (isf_{t_j} is the inverse sentence frequency of term t_j , and tf_{ij} is the term frequency of term t_j in sentence s_i). A^T is the transpose of A .

Table 3 compares the ROUGE results of *HyperSum* with query sensitive LexRank and HITS on DUC 2006 data set. We can see that *HyperSum* outperforms query sensitive LexRank and HITS in all three measures. It is above query sensitive LexRank by 7.22% of ROUGE-2 and 3.77% of ROUGE-SU4, respectively. As for HITS, the improvements are 17.22% and 11.42% with respect to ROUGE-2 and ROUGE-SU4. These achievements are significant since in the DUC evaluation except for the best system, the performance of the other top ranking systems are very close, i.e. the second best system is only 1.60% above the fourth best system on ROUGE-2 and 0.86% on ROUGE-SU4. Then the comparison is extended to DUC 2005 and 2007 data sets, which are shown in Table 4. We also can see that *HyperSum* is better than query sensitive LexRank and HITS with respect to both ROUGE-2 and ROUGE-SU4 scores.

6.6. Comparison with DUC systems

Next, we compare *HyperSum* with the DUC participating systems. 34 systems have been submitted for evaluation in DUC 2006. To provide a global picture of the evaluation results, we provides (1) the worst human summarizer (denoted by H), which reveals the gap between the system generated summaries and the human generated summaries; (2) top nine ranked DUC participating systems and one trail system (denoted by S24, S11 and so on); (3) the average ROUGE scores (denoted by SYS AVG); and (4) the NIST baseline, which randomly picks a document and selects the leading sentences. Then we can easily locate the position of our proposed approach within the ranking. Clearly, *HyperSum* ranks the first in the DUC 2006, as illustrated in Table 5.

To further examine the adaptability of the proposed approach, we extend the previous experiments to the DUC 2005 and DUC 2007 data sets. The parameters are set to the ones which have achieved the best performance on the DUC 2006 data set, i.e. $a = 0.98$, $b = 1.5$. For both DUC 2005 and DUC 2007, 31 systems have participated in their evaluations. It is obvious that, *HyperSum* ranks the first with respect to ROUGE-2 in the DUC 2005, as illustrated in Table 6. As for DUC 2007, Table 7 shows that *HyperSum* ranks the fifth. Since the details of text processing can influence the summarization performance, we find that all the systems (i.e. S15, S29, S4 and S24) better than ours have put much emphasis on the text preprocessing and summary post-processing such as sentence trimming and entity dereference [20]. In contrast, since the

Table 3
Comparison with conventional graph based models on DUC 2006.

	ROUGE-2	ROUGE-SU4
<i>HyperSum</i> ($a = 0.98$, $b = 1.5$)	0.09569 (0.08722–0.10404)	0.15182 (0.14424–0.15899)
Query Sensitive LexRank ($d = 0.15$)	0.08924 (0.08015–0.09720)	0.14630 (0.13853–0.15378)
HITS	0.08163 (0.07357–0.08924)	0.13625 (0.12787–0.14419)

Table 4
Comparison with conventional graph based models on DUC 2005 and 2007.

Data	ROUGE-2	ROUGE-SU4
2005		
<i>HyperSum</i> ($a = 0.98$, $b = 1.5$)	0.07291 (0.06424–0.08086)	0.13087 (0.12226–0.13872)
Query Sensitive LexRank ($d = 0.15$)	0.07102 (0.06316–0.07829)	0.12641 (0.11834–0.13410)
HITS	0.068183 (0.05925–0.07641)	0.12245 (0.11318–0.13078)
2007		
<i>HyperSum</i> ($a = 0.98$, $b = 1.5$)	0.11174 (0.10368– 0.11972)	0.16587 (0.15821–0.17417)
Query Sensitive LexRank ($d = 0.15$)	0.11025 (0.10204–0.11824)	0.16501 (0.15731–0.17271)
HITS	0.10470 (0.09580–0.11327)	0.15733 (0.14852–0.16574)

Table 5
Comparison with DUC 2006 systems.

	ROUGE-2	ROUGE-SU4
H	0.13260	0.18385
...		
HyperSum	0.09569	0.15182
S24	0.09558	0.15529
S15	0.09097	0.14733
S12	0.08987	0.14755
S8	0.08954	0.14607
S23	0.08792	0.14486
S28	0.08700	0.14522
S31	0.08576	0.14381
S2	0.08536	0.14094
S33	0.08444	0.14483
...		
S11	0.02834	0.06394
SYS AVG	0.07463	0.13021
NIST baseline	0.04947	0.09788

Table 6
Comparison with DUC 2005 systems.

	ROUGE-2	ROUGE-SU4
H	0.88593	0.14843
...		
HyperSum	0.07291	0.13087
S15	0.07251	0.13163
S17	0.07174	0.12972
S10	0.06984	0.12525
S8	0.06963	0.12795
S4	0.06858	0.12773
S5	0.06750	0.12324
S11	0.06426	0.12551
S14	0.06349	0.11763
S16	0.06326	0.11897
...		
S23	0.02564	0.05569
SYS AVG	0.05842	0.11205
NIST baseline	0.04026	0.08716

Table 7
Comparison with DUC 2007 systems.

	ROUGE-2	ROUGE-SU4
H	0.17528	0.21892
...		
S15	0.12448	0.17711
S29	0.12028	0.17074
S4	0.11887	0.16999
S24	0.11793	0.17593
HyperSum	0.11174	0.16587
S13	0.11172	0.16446
S20	0.10879	0.15844
S23	0.10810	0.16280
S7	0.10795	0.15990
S3	0.10660	0.15991
...		
S16	0.03813	0.07385
SYS AVG	0.09597	0.14884
NIST baseline	0.06039	0.10507

focus of this work is to address the issues of text hypergraph and semi-supervised hypergraph based ranking, we do not involve much preprocessing and post-processing, which will definitely boost the overall performance of our system and need further research.

7. Conclusions

In this paper, we propose a novel query-oriented summarization approach *HyperSum* which incorporates the text hypergraph into the semi-supervised sentence ranking framework. In addition to the pairwise relationship in existing graph based models, a text hypergraph can integrate more group relations among multiple sentences. Under the assumption that a given topic or event usually covers several subtopics each of which is described by a group of sentences, in this paper we mainly consider the subtopic relationship as well as the pairwise ones which are formulated as hyperedges in a hypergraph. Through the structure of the hypergraph, the query relevance is then recursively propagated from the labeled query to the unlabeled sentences. In *HyperSum*, selection of similarity metric and construction of subtopics are two main problems. To select an appropriate similarity metric, we compare three widely used metrics: *Cosine*, *Jaccard* and *Euclidean* and find *Cosine* can achieve a better summarization performance than the other two metrics. At the same time, various clustering techniques including *K-means*, agglomerative hierarchical clustering, divisive hierarchical clustering, latent Dirichlet allocation (LDA) and DBSCAN are compared and a modified DBSCAN is finally proposed since it does not require a predefined cluster number and has the capability of removing noise. The experiments are conducted on three years of DUC (2005, 2006 and 2007) data sets, verifying that *HyperSum* can be seen as one of the best systems in each year's evaluation. We believe that *HyperSum* can be expected to reach a better performance in the query-oriented summarization task if more relations among sentences can be discovered and incorporated in a hypergraph. In our future work, we will further investigate the discovery and computation of various relationship existing among sentences.

Acknowledgments

The work described in this paper was in part supported by NSFC programs (Nos.: 90920011 and 61273278), National Key Technology R&D Program (No.: 2011BAH10B04-03), and National High Technology Research and Development Program of China (863 Program) (No. 2012AA011101). Also we would like to thank the anonymous reviewers for their valuable comments.

References

- [1] C. Blake, A comparison of document, sentence, and term event spaces, in: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, 2006, pp. 601–608.
- [2] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, *The Journal of Machine Learning Research* (2003) 993–1022.
- [3] X. Cai, W. Li, A spectral analysis approach to document summarization: clustering and ranking sentences simultaneously, *Information Sciences* 181 (18) (2011) 3816–3827.
- [4] A. Corduneanu, T. Jaakkola, Distributed information regularization on graphs, *Advances in Neural Information Processing Systems* 17 (2005) 297–304.
- [5] L. Ding, A. Yilmaz, Image segmentation as learning on hypergraphs, *International Conference on Machine Learning and Applications* (2008) 247–252.
- [6] P. Du, J. Guo, J. Zhang, X. Cheng, Manifold ranking with sink points for update summarization, in: Proc. of CIKM'2010, 2010, pp. 1757–1760.
- [7] G. Erkan, D. Radev, LexRank: graph-based centrality as salience in text summarization, *Journal of Artificial Intelligence Research* 22 (2004) 457–479.
- [8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of KDD 1996, 1996, pp. 226–231.
- [9] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, *Journal of the ACM* 30 (3) (1983) 514–550.
- [10] E.-H. Han, G. Karypis, V. Kumar, B. Mobasher, Hypergraph-based clustering in high-dimensional data sets: a summary of results, *Bulletin of the Technical Committee on Data Engineering* 21 (1) (1998) 15–22.
- [11] G. Karypis, V. Kumar, Multilevel *k*-way hypergraph partitioning, *VLSI Design* 11 (3) (2000) 285–300.
- [12] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, *Journal of the ACM* 46 (5) (1999) 604–632.
- [13] W. Li, F. Wei, Q. Lu, Y. He, PNR²: ranking sentences with positive and negative reinforcement for query-oriented update summarization, in: Proc. of COLING '08, 2008, pp. 489–496.
- [14] X. Li, L. Du, Y. Shen, Graph-based marginal ranking for update summarization, in: Proc. of SDM 2011, 2011, pp. 486–497.
- [15] C.-Y. Lin, E.H. Hovy, The automated acquisition of topic signature for text summarization, in: Proceedings of COLING 2000, 2000, pp. 495–501.
- [16] C.-Y. Lin, E.H. Hovy, Automatic evaluation of summaries using *n*-gram co-occurrence Statistics, in: Proceedings of HLT-NAACL 2003, 2003, pp. 71–78.
- [17] J. Otterbacher, G. Erkan, D. Radev, Using random walks for question-focused sentence retrieval, in: Proceedings of HLT/EMNLP 2005, 2005, pp. 915–922.
- [18] Y. Ouyang, W. Li, S. Li, Q. Lu, Applying regression models to query-focused multi-document summarization, *Information Processing and Management* 47 (2) (2011) 227–237.
- [19] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web, Technical Report, Stanford Digital Libraries, 1998.
- [20] R.K.P. Pingali, V. Varma, IIT Hyderabad at DUC 2007, in: Document Understanding Conference, 2007. <<http://duc.nist.gov>>.
- [21] D.R. Radev, H.Y. Jing, M. Stys, D. Tam, Centroid-based summarization of multiple documents, *Information Processing and Management* 40 (2004) 919–938.
- [22] X. Wan, J. Yang, J. Xiao, Manifold-ranking based topic-focused multi-document summarization, in: Proc. of IJCAI 2007, 2007, pp. 2903–2908.
- [23] X. Wan, J. Yang, J. Xiao, Towards iterative reinforcement approach for simultaneous document summarization and keyword extraction, in: Proceedings of ACL 2007, 2007, pp. 552–559.
- [24] X. Wan, J. Yang, Multi-document summarization using cluster-based link analysis, in: Proceedings of SIGIR 2008, 2008, pp. 299–306.
- [25] B. Wang, Z. Li, M. Li, Automatic refinement of keyword annotations for web image search, in: Proc. of MMM 2007, 2007, pp. 259–268.
- [26] F. Wang, C. Zhang, Label propagation through linear neighborhoods, in: Proceedings of ICML 2006, 2006, pp. 55–67.
- [27] X. Wang, J. Chen, C. Zhu, User-specific video summarization, in: 2011 International Conference on Multimedia and Signal Processing, vol. 1, 2011, pp. 213–219.
- [28] F. Wei, W. Li, Q. Lu, Y. He, Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization, in: Proceedings of SIGIR 2008, 2008, pp. 283–290.
- [29] F. Wei, W. Li, Q. Lu, Y. He, Applying two-level reinforcement ranking in query-oriented multi-document summarization, *Journal of the American Society for Information Science and Technology* 60 (10) (2009) 2119–2131.
- [30] F. Wei, W. Li, Q. Lu, Y. He, A document-sensitive graph model for multi-document summarization, *Knowledge and Information Systems* 22 (2) (2010) 245–259.

- [31] K. Wong, M. Wu, W. Li, Extractive summarization using supervised and semi-supervised learning, in: Proceedings of COLING 2008, pp. 985–992.
- [32] S. Ye, T.-S. Chua, M.-Y. Kan, L. Qiu, Document concept lattice for text understanding and summarization, *Information Processing and Management* 43 (6) (2007) 1643–1662.
- [33] H. Zha, Generic summarization and key phrase extraction using mutual reinforcement principle and sentence clustering, in: Proceedings of SIGIR 2002, 2002, pp. 113–120.
- [34] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: *Advances in Neural Information Processing Systems* 16, Cambridge, MA, 2004, pp. 321–328.
- [35] D. Zhou, J. Huang, B. Schölkopf, Beyond Pairwise Classification and Clustering Using Hypergraphs, MPI Technical Report, 143, Tübingen, Germany, 2005.
- [36] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised Learning using Gaussian Fields and Harmonic Functions, in: Proceedings of ICML 2003, 2003, pp. 912–919.