Keyphrase Extraction Using Knowledge Graphs

Wei Shi[1], Weiguo Zheng[1]✉, Jeffrey Xu Yu[1], Hong Cheng[1], and Lei Zou[2]

[1] The Chinese University of Hong Kong
{shiw,wgzheng,yu,hcheng}@se.cuhk.edu.hk
[2] Peking University
zoulei@pku.edu.cn

Abstract. Extracting keyphrases from documents automatically is an important and interesting task since keyphrases provide a quick summarization for documents. Although lots of efforts have been made on keyphrase extraction, most of the existing methods (the co-occurrence based methods and the statistic-based methods) do not take semantics into full consideration. The co-occurrence based methods heavily depend on the co-occurrence relations between two words in the input document, which may ignore many semantic relations. The statistic-based methods exploit the external text corpus to enrich the document, which introduce more unrelated relations inevitably. In this paper, we propose a novel approach to extract keyphrases using knowledge graphs, based on which we could detect the latent relations of two keyterms (i.e., noun words and named entities) without introducing many noises. Extensive experiments over real data show that our method outperforms the state-of-art methods including the graph-based co-occurrence methods and statistic-based clustering methods.

1 Introduction

The continuously increasing text data, such as news, articles, and papers, make it urgent to design an effective and efficient technique to extract high-quality keyphrases automatically since keyphrases help us to have a quick knowledge of the text. Keyphrase extraction also provides useful resources for text clustering [9], text classification [30], and document summarization [28]. There are two main types of studies on keyphrase extraction, i.e., supervised and unsupervised. Majority of the supervised methods regard keyphrase extraction as a binary classification task [26], [12], [29], [13], [15], and take some features, such as term frequency-inverse document frequency (*tf-idf*) and the position of the first occurrence of a phrase, as the inputs of a Naive Bayes classifier [24]. However, a binary classifier fails to rank the phrases since each candidate phrase is classified independently with others. More importantly, supervised methods demand a lot of training data, i.e., manually labeled keyphrases. This is extremely expensive and time-consuming in domain-specific scenarios. In order to reduce manpower, investigating comparative unsupervised methods is highly desired. Thus, we focus on studying unsupervised methods to extract keyphrases from a single input document (e.g., news and article).

The existing unsupervised approaches can be divided into two categories, i.e., co-occurrence based methods and statistic-based methods, as shown in Table 1. The co-occurrence based methods, e.g., *SW* [11], *TextRank* [21], *ExpandRank* [27], *CM* [5],

Table 1. Overview of the approaches

Category	Methods	Information Sources
Co-occurrence based	*CM* [5], *SW* [11], *TextRank* [21], *ExpandRank* [27]	*Input documents*
Statistic-based	*Wan'07* [28], *SC* [18], *TPR* [17]	*External text corpus, e.g., Wikipedia article*
Semantics-based	*Our method*	*Knowledge graphs, e.g., DBpedia*

build a word co-occurrence graph exploiting the word co-occurrence relations that are obtained from the input document, and then apply some ranking algorithms such as PageRank [23] and betweenness [7] on the graph to get the ranking score of each word. Based on the ranking score, the *top-k* phrases are returned as keyphrases. The statistic-based methods, e.g., *Wan'07* [28], *SC* [18], and *TPR* [17], explore some external text corpus to assist keyphrase extraction. For example, *SC* [18] uses the Wikipedia [1] articles, based on which several statistical distances can be computed, such as cosine similarity, Euclidean distance, Pointwise Mutual Information (PMI), and Normalized Google Similarity Distance (NGD) [6]. It first clusters candidate words based on the statistical distance, and then selects the exemplar terms. Finally, the candidate phrases that contain exemplar terms are selected as keyphrases.

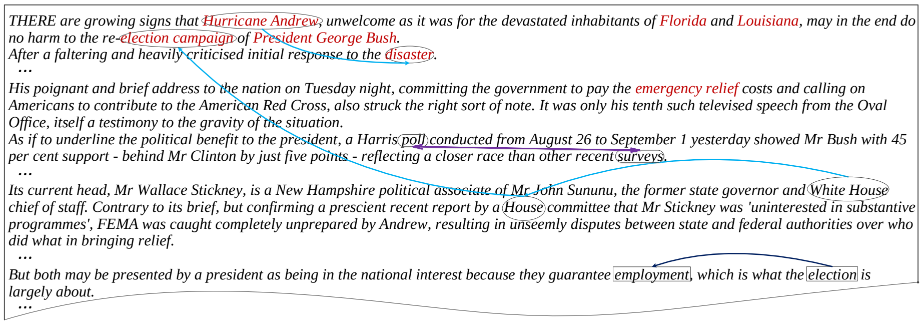
However, the existing methods suffer from two drawbacks: *information loss* and *information overload*.

– Information Loss. The co-occurrence based methods heavily depend on the co-occurrence relations between two words. If two words never occur together within a predefined window size in an input document, there will be no edges to connect them in the built co-occurrence graph even though they are semantically related. Furthermore, as reported in [11], the performance improves little by increasing the window size w if w exceeds a small threshold (w ranges from 3 to 9).
– Information Overload. The statistic-based methods exploit external text corpus to enrich the input document. Nevertheless, the real meanings of words in the document may be overwhelmed by the large amount of introduced external texts. Furthermore, they can only acquire very limited useful knowledge about the words in the input document since they just use the statistical information of two words in the external texts actually.

Running Example. *Figure 1 shows a document from the dataset DUC2001, where the keyphrases are labeled in red font.*

In the co-occurrence based methods, the words adjacent to more other words tend to rank higher. Let us consider two candidate phrases "disaster" and "Hurricane Andrew". They do not occur together within a window if the window size is 10. Thus no edges will be induced to connect them in the co-occurrence graph, which indicates they are not highly related while "Hurricane Andrew" is an instance of "disaster". Hence, "disaster" is not delivered as a keyphrase using the co-occurrence based methods, e.g., TextRank and ExpandRank.

The statistic-based methods directly map a word to the words with the same surface form in the external text resources. The real meaning of a word in the input document may be overwhelmed by the extended corpus. For instance, the most common meaning of "house" in Wikipedia is "House Music", which is different from the meaning "United States House of Representatives" in the example document. Hence, statistic-based methods fail to exclude the wrong senses and result in bad similarity relations.



THERE are growing signs that Hurricane Andrew, unwelcome as it was for the devastated inhabitants of Florida and Louisiana, may in the end do no harm to the re-election campaign of President George Bush.
After a faltering and heavily criticised initial response to the disaster.
...
His poignant and brief address to the nation on Tuesday night, committing the government to pay the emergency relief costs and calling on Americans to contribute to the American Red Cross, also struck the right sort of note. It was only his tenth such televised speech from the Oval Office, itself a testimony to the gravity of the situation.
As if to underline the political benefit to the president, a Harris poll conducted from August 26 to September 1 yesterday showed Mr Bush with 45 per cent support - behind Mr Clinton by just five points - reflecting a closer race than other recent surveys.
...
Its current head, Mr Wallace Stickney, is a New Hampshire political associate of Mr John Sununu, the former state governor and White House chief of staff. Contrary to its brief, but confirming a prescient recent report by a House committee that Mr Stickney was 'uninterested in substantive programmes', FEMA was caught completely unprepared by Andrew, resulting in unseemly disputes between state and federal authorities over who did what in bringing relief.
...
But both may be presented by a president as being in the national interest because they guarantee employment, which is what the election is largely about.
...

Fig. 1. An example of input document.

In this paper, we propose a novel method to extract keyphrases by considering the underlying semantics. Relying on semantics, we can address the problems above. (1) Using semantics, we can find the latent relations between two keyterms (see Sect. 2.1). As shown in the example document in Fig. 1, although the distance between "disaster" and "Hurricane Andrew" in the document is large, it is easy to add an edge between them if we know "Hurricane Andrew" is an instance of "disaster" according to their semantic relation. (2) In order to avoid introducing many noisy data, we attempt to include the useful information by incorporating semantics. For example, through entity linking, our method could select "United States House of Representatives" as the proper meaning of "house" in the example document in Fig. 1 and only bring in the corresponding semantic relations, thus excluding the noisy relations caused by "House Music".

In order to incorporate the semantics for keyphrase extraction, we resort to knowledge graphs in this paper. A knowledge graph, e.g., DBpedia [3], captures lots of entities, and describes the relationships between the entities. Note that two earlier work communityRank [8] and SemanticRank [25] utilize the Wikipedia page linkage as the external knowledge. However, they just use the link information in a coarse-grained statistical way. Different from the work above, we propose to incorporate semantics into keyphrase extraction by adopting the structure of knowledge graph (e.g., DBpedia).

The major contributions of this paper are summarized as follows:

- We are the first to use the structure of knowledge graphs to provide semantic relations among keyterms in keyphrase extraction task.
- We propose a systematic framework that integrates topic clustering and graph ranking for keyphrase extraction.
- By considering the real observations, we give a novel and reasonable definition of keyphrases, i.e., the important noun phrases of a document.
- Extensive experiments over real data show that our method achieves better performance compared with the classic co-occurrence based methods and statistic-based methods.

The rest of this paper is organized as follows. Section 2 formulates the task of the work and gives an overview of our method. To cover more topics of a document, we perform the topic clustering in Sect. 3. Section 4 studies how to incorporate knowledge graphs to model the keyterms. Followed by the process of generating keyphrases in Sect. 5. Section 6 reports experiments on real data. In Sect. 7, we review related work in recent years. Finally, we conclude this work in Sect. 8.

2 Problem Definition & Framework

In this section, we first formulate our problem and then give the overview of our methods. Table 2 lists the notations used in the paper.

Table 2. Notations

Notation	Definition
d	Document
$G = (V, E, LA)$	Knowledge graph
$P = \{p_1, p_2, ..., p_n\}$	A group of noun phrases
$KT = \{kt_1, kt_2, ..., kt_m\}$	A group of keyterms
$C = \{c_1, c_2, ..., c_r\}$	Clusters of KT
$AN = \{v_{a_1}, v_{a_2}, ..., v_{a_t}\}$	A group of anchor nodes
$EV \subset V$	A group of expanded nodes
L	The path between two anchor nodes
$EV.L$	The expanded nodes on the path L
$KG_h(AN)$	h-hop keyterm graph induced by AN
$\mu(EV.L, AN)$	The semantic relatedness between $EV.L$ and AN
τ	The semantic relatedness threshold

2.1 Problem Formulation

A ***named entity*** is a real-world object such as location, organization, person and so on. To name a few, "George Bush", "Florida", "Red Cross", "Gulf of Mexico". A ***noun phrase*** is composed of continuous noun words or named entities. For example, "president George Bush", "Hurricane Andrew", "New York City" are all noun phrases.

Noun words and *named entities* in the document are called ***Keyterms***. For example, "hurricane", "Florida", "George Bush" are all keyterms.

Definition 1 (Keyphrase). *A keyphrase consists of keyterms and is highly relevant to the topics of a document.*

For instance, "president george bush", "hurricane andrew", "florida", "louisiana", "election campaign", "emergency relief" and "disaster" are the keyphrases of the example document in Fig. 1.

Remark. In this paper, we specify that a keyphrase should be a noun phrase. The reasons are listed as follows: (1) Noun phrases are more substantive, while adjectives always serve as the descriptive function. Table 3 presents some phrases with and without adjectives, while the meanings of phrases do not change much. Moreover, the rate of noun phrases in the manually labeled keyphrases by Wan et al. is 64% [27]. (2) In general, different annotators may label different groups of keyphrases according to their understanding. But they always reach high agreement on noun phrases, which is verified through our experiments (see Sect. 6.3).

Problem 1 (Keyphrase Extraction). Keyphrase extraction is the task of extracting a group of keyphrases from a document with good coverage of the topics.

Table 3. Phrases with and without adjectives

with Adjectives	w/o Adjectives
widespread destruction	destruction
severe damage	damage
serious injury	injury
massive aircraft	aircraft
large immigrant population	immigrant population
small twister	twister

2.2 Framework of Our Method

Figure 2 shows our framework. Given a document, we first select the keyterms. Then, we apply clustering algorithms on keyterms based on the semantic similarity which aims at covering more topics of the document. The keyterms in each cluster are linked to entities in the knowledge graph. For each cluster, taking the mapped entities as anchor nodes, we can detect the latent keyterm-keyterm relations in the input document through extracting the *h-hop* keyterm graph (please refer to Definition 2) from the knowledge graph. Then, we apply Personalized PageRank (PPR) [10] on each keyterm graph and obtain the ranking score of each keyterm. To distinguish the importance of different clusters, we regard the centers of all the clusters as keyterms and form a new cluster and then rank these centers using the similar method as keyterms. Finally, the keyphrases are generated based on the ranking scores of keyterms.

Topic clustering. In general, a document consists of multiple topics. Thus we propose a clustering method to cover more topics. Specifically, the keyterms are clustered according to the semantic similarity. Then we try to generate keyphrases from each cluster. Different from the existing methods that employ single word or n-gram as the clustering element, we use keyterms (i.e., the noun words and named entities) in this paper.

Keyterm graph construction. For the keyterms in each cluster, we build a keyterm graph by exploiting the structure of the knowledge graph. The keyterm graph describes the semantic relations among keyterms.

Graph-based ranking and phrase generation. We can adopt the ranking algorithms, such as PPR and SimRank [14], to compute the importance of each keyterm. Then we can get the ranking score of each candidate phrase. The k candidate phrases with the largest scores are delivered as keyphrases.

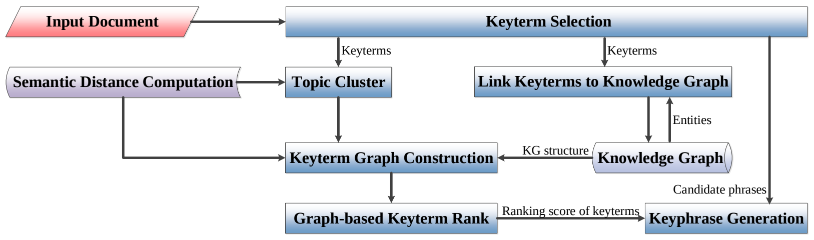


Fig. 2. System framework.

3 Topic Clustering

Since a document always has more than one topics, keyphrases should represent all these topics. Hence we propose to cluster the keyterms according to semantics, and then generate the keyphrases based on the clusters.

3.1 Keyterm Selection

The existing method *SC* [18] also proposes a clustering approach to cover the topics of a document. However, it clusters all words of the document without distinguishing the parts of speech, which may introduce many noisy data. Mostly, noun words and named entities describe the key points of a document. Hence, we propose to take noun words and named entities as keyterms. Since the knowledge graph consists of entities and relations, it is easy to integrate the knowledge graph into keyphrase extraction by mapping keyterms to entities. To extract the keyterms, we resort to the existing NLP tools, e.g., Stanford CoreNLP [20].

3.2 Keyterm Clustering

In this paper, we apply k-means clustering algorithm [19] to divide the keyterms into r clusters. K-means is one of the simplest unsupervised learning algorithms to solve the well-known clustering problem. It partitions m objects into r clusters such that each object belongs to the closest cluster in terms of mean distance.

For each cluster, we select the keyterm which is nearest to the corresponding centroid and has candidate entities in the knowledge graph G as the final centroid. In order to convey the semantics, we use Google Word2vec [22] to compute the semantic distance of two keyterms. If a keyterm kt is a named entity that consists of multiple words, we compute the vector representation of kt according to the words that are contained in kt as the work [16] does.

Example 1. Figure 3 shows the clustering result on the example document, where r is set to be 7. As the color deepened, the corresponding cluster is more important and relevant to the main topics of the example document. Keyterms in blue are the cluster centroids.

4 Incorporating Knowledge Graphs into Keyphrase Extraction

As discussed above, considering the semantics, we can detect more relations, which can improve the ranking results. In this paper, we propose to adopt knowledge graph to assist keyphrase extraction. However, it is unclear how to integrate knowledge graph in a better way since there is no such related study. We try to give a solution in this work.

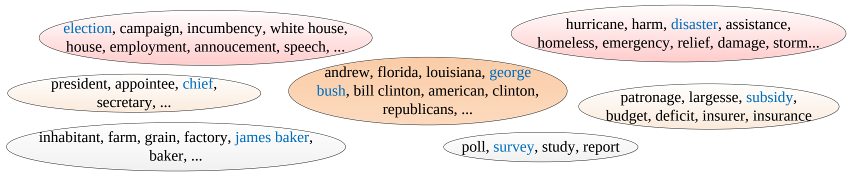

election, campaign, incumbency, white house, house, employment, annoucement, speech, ...
hurricane, harm, disaster, assistance, homeless, emergency, relief, damage, storm...
president, appointee, chief, secretary, ...
andrew, florida, louisiana, george bush, bill clinton, american, clinton, republicans, ...
patronage, largesse, subsidy, budget, deficit, insurer, insurance
inhabitant, farm, grain, factory, james baker, baker, ...
poll, survey, study, report

Fig. 3. Clustering result when cluster number $r = 7$.

4.1 Keyterm Linking

Given a knowledge graph G, the keyterm linking task is to link keyterms to G, i.e., find the mapping entity in G for each keyterm. A keyterm, regarded as a surface form, usually has multiple mapping entities. For example, the keyterm "Philadelphia" may refer to a city or a film. There have been many studies that focus on sense disambiguation. Most of the existing methods demand that each entity in G has a page p that describes the entity. Then the semantic relatedness between p and the input document d could be computed, e.g., computing the cosine similarity between them. The entity that corresponds to the page with the largest similarity score is selected as the mapping entity.

Since the surface forms are keyterms, we build a mapping dictionary for keyterms, which materializes the possible mappings for a keyterm kt. Each possible mapping is assigned a prior probability. To compute the prior probability, we resort to Wikipedia articles. For each linkage (i.e., the underlined phrase that is linked to a specific page) in Wikipedia, we can get a pair of surface form and entity. Then we can get the probability of each candidate entity e_i by calculating the proportion of its co-occurrence number with kt over the total occurrence number of kt as shown in (1).

$$Pr\{(kt, e_i)\,|kt\} = \frac{T\,(kt, e_i)}{T\,(kt)}, \tag{1}$$

where $T\,(kt, e_i)$ is the co-occurrence number of kt and e_i, and $T\,(kt)$ is the occurrence number of kt. Given a keyterm kt, we explore the dictionary to find its candidate mappings. For an entity in Wikipedia, we use the corresponding Wikipedia page as its context. For entities in DBpedia, we use the dataset "long_abstracts_en.ttl" as the context. If the knowledge graph has page descriptions, we can integrate the semantic relatedness (we adopt cosine similarity between the context of an entity and the input document in our experiments) and the prior probability to decide the correct mapping.

Example 2. Consider the keyterm "house" in the example document, there are several candidate entities in Wikipedia, including "House music", "United States House of Representatives", "House", "House system". Our method successfully selects "United States House of Representatives" as the appropriate entity.

4.2 Keyterm Graph Construction

After the keyterm linking, we obtain the mapping entities of keyterms, which are also called *anchor nodes*. Next we build a *h-hop* keyterm graph to capture the semantic relations among keyterms for each cluster. Consider a knowledge graph $G = (V, E, LA)$, where V is the set of nodes, E is the set of edges, and LA is the set of node labels. Let AN denote the set of anchor nodes, i.e., $AN = \{v_{a_1}, v_{a_2}, ..., v_{a_t}\} \subset V$. Definition 2 describes the *h-hop* keyterm graph.

Definition 2 (H-hop Keyterm Graph). *Given the set of anchor nodes AN in the knowledge graph G, the h-hop keyterm graph, denoted by $KG_h\,(AN)$, is the subgraph of G that includes all paths of length no longer than h between v_{a_i} and v_{a_j}, where $v_{a_i}, v_{a_j} \in AN$ and $i \neq j$.*

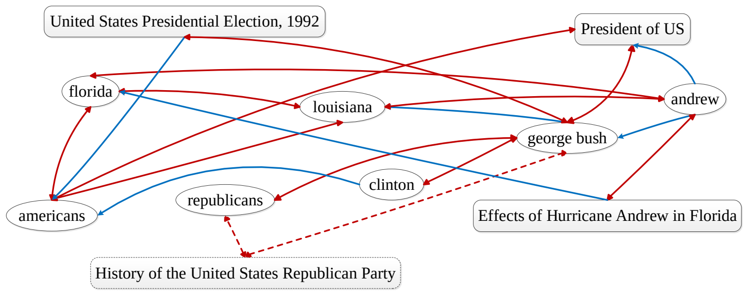

United States Presidential Election, 1992
President of US
florida
louisiana
andrew
george bush
clinton
republicans
americans
Effects of Hurricane Andrew in Florida
History of the United States Republican Party

Fig. 4. H-hop keyterm graph.

Each path in the *h-hop keyterm graph* describes the relation between two anchor nodes. Since the anchor nodes are the mapping entities for keyterms, the *h-hop keyterm graph* models the semantic relations among the keyterms. The newly introduced nodes in $KG_h(AN)$ excluding the anchor nodes are the *expanded nodes*, denoted by EV. In order to eliminate the noisy nodes, we need to refine the *h-hop keyterm graph* by considering the semantics of the whole set of anchors. Algorithm 1 shows the process of constructing the *h-hop keyterm graph*. For $\forall v_{a_i} \in AN$, we apply Breadth-first search (BFS) to extract the paths no longer than h between v_{a_i} and v_{a_j} ($\forall v_{a_j} \in AN$ and $i \neq j$). Next we remove the paths which are less related with AN. Let us consider the path L between two anchor nodes in $KG_h(AN)$. The set of expanded nodes in L is denoted as $EV.L$. Then we compute the semantic relatedness $\mu(EV.L, AN)$ between $EV.L$ and AN. If $\mu(EV.L, AN)$ is less than a threshold τ, L is removed from the *h-hop keyterm graph*. To compute the semantic relatedness $\mu(EV.L, AN)$, we utilize Word2vec and compute cosine similarity between the vector representations of $EV.L$ and AN. The complexity of Algorithm 1 is $\mathcal{O}(N \cdot (\frac{|E|}{|V|})^h)$, where N is the number of anchor nodes.

Example 3. Figure 4 shows part of the h-hop keyterm graph for the cluster in the middle of Fig. 3, where ellipses are anchor nodes and rectangles are expanded nodes. As the dotted bordered rectangle shows, although "History of the United States Republican Party" connects with "republicans" and "george bush", it is less associated with other anchor nodes. Hence, it is removed in the expansion process.

After incorporating the knowledge graph, we can add some relations that are derived from the input document to the keyterm graph. Consider two anchor nodes v_{a_i} and

Algorithm 1 *H-hop Keyterm Graph* Construction

Input: Knowledge graph $G = (V, E, LA)$, anchor nodes AN, semantic relatedness threshold τ
Output: *H-hop keyterm graph*

```
1: for $v_{a_i} \in AN$ do
2:     for $v_{a_j} \in AN$ and $i \neq j$ do
3:         $Path(v_{a_i}, v_{a_j}) \leftarrow$ extract all the simple paths no longer than $h$ between $v_{a_i}$ and $v_{a_j}$
4:         for $L \in Path(v_{a_i}, v_{a_j})$ do
5:             $\mu(EV.L, AN) \leftarrow$ compute the semantic relatedness between $EV.L$ and $AN$
6:             if $\mu(EV.L, AN) < \tau$ then
7:                 remove $L$ from $Path(v_{a_i}, v_{a_j})$
8: return the subgraph consisting of $Path(v_{a_i}, v_{a_j})$
```

v_{a_j}. If their corresponding keyterms kt_i and kt_j occur in the same window, an edge is added between v_{a_i} and v_{a_j}. For each cluster, we build a keyterm graph. To measure the importance of each cluster, we can use the centroids of all clusters as the input keyterms, and build a corresponding keyterm graph for all the centroids using the same method.

5 Keyphrase Generation

5.1 Keyterm Ranking

After constructing keyterm graphs, the next focus is how to measure the importance of different keyterms. Note that our keyterm graph is a semantic graph, where edges reflect semantic relations between keyterms. But the existing methods show that term frequency is a very important feature of keyphrase. Hence, we need a graph centrality method that ranks keyterms by considering both the semantic structure of the keyterm graph and the frequency of keyterms. Although PageRank ranks nodes using the graph structure, the jump probability of PageRank is normalized, which indicates that it fails to reflect the difference of node features. In contrast, Personalized PageRank (PPR), which is a variant of PageRank, assigns a biased jump probability to each node and ranks graph nodes based on the graph structure. Therefore, it is able to reflect the keyterm frequency by assigning the jump probability according to the keyterm frequency. We formulate the PPR based on keyterm frequency as follows:

$$\mathrm{PPR}\left(v_i\right) = \lambda \times \sum_{v_j \to v_i} \frac{\mathrm{PPR}\left(v_j\right)}{\mathrm{outdegree}\left(v_j\right)} + (1-\lambda) \times p\left(v_i\right), \tag{2}$$

where $\mathrm{PPR}\left(v_i\right)$ is the PPR score of v_i, $\mathrm{outdegree}\left(v_j\right)$ is the outdegree of v_j, $p\left(v_i\right)$ is the jump probability of v_i and λ is a damping factor which is usually set to 0.85 [23]. The jump probability $p\left(v_i\right)$ is computed according to (3), where N is the number of nodes.

$$p\left(v_i\right) = \frac{\mathrm{tf}\left(v_i\right)}{\sum\limits_{i=1}^{N} \mathrm{tf}\left(v_i\right)}, i = 1, ..., N. \tag{3}$$

Since the expanded nodes do not have matching keyterms in the document, their jump probabilities are set to be 0. We perform PPR on the keyterm graph that corresponds to each cluster and cluster centroids. Then we obtain the ranking scores of the corresponding anchor nodes, including the PPR score of each cluster centroid.

5.2 Keyphrase Generation

With the ranking scores of keyterms, next we extract the keyphrases. The main idea is to generate candidate phrases first and then rank them based on the scores of keyterms.

The existing phrase generation methods [18], [27] adopt the syntactic rule $(\mathrm{JJ})*(\mathrm{NN|NNS|NNP})+$, where "JJ" is an adjective, "NN", "NNS", and "NNP" are nouns, "*" and "+" mean zero or more adjectives and at least one noun word should be contained in the candidate phrase. For instance, given a sentence "*Federal/NNP Emergency/NNP Management/NNP Agency/NNP has/VBZ become/VB the/DT ultimate/JJ* *patronage/NN backwater/NN*", using this rule, candidate phrases "federal emergency management agency" and "ultimate patronage backwater" are generated.

We propose to extract important noun phrases as keyphrases in Sect. 2.1. Hence, we design a rule by removing the adjectives, i.e., $(\mathrm{NN}|\mathrm{NNS}|\mathrm{NNP})+$. Following this rule, except named entities, all the candidate phrases are a chain of continuous noun words. For the example sentence, candidate phrases "federal emergency management agency" and "patronage backwater" are generated.

After generating the candidate noun phrases, we propose a method to compute the ranking score of a candidate phrase by combining the scores of keyterms contained in it. Since keyterms in different clusters cannot be compared directly, we put forward a measurement to compare keyterms from different clusters. In the semantic subgraph construction step, we build a keyterm graph for cluster centroids. Similar to the keyterms in the same cluster, different cluster centroids can be compared as well. It makes sense that if a cluster contains many keyterms, i.e., tf is larger, the cluster is more important. So we combine the ranking score of a cluster centroid and the total tf of the cluster to form a comprehensive score for the cluster, as shown in (4). Next we combine the PPR score of a keyterm and the ranking score of its cluster to form a global utility score for the keyterm, as shown in (5).

$$\mathrm{SC}\left(c_i\right) = \alpha \times \mathrm{PPR}\left(c_i\right) + (1-\alpha) \times \frac{\sum_{j=1}^{t} \mathrm{tf}\left(kt_{ij}\right)}{\log_2 t}, \forall c_i \in C, kt_{ij} \in c_i. \tag{4}$$

$$\mathrm{Score}\left(kt_{ij}\right) = \mathrm{PPR}\left(kt_{ij}\right) \times \mathrm{SC}\left(c_i\right), \quad \forall kt_{ij} \in c_i, \forall c_i \in C. \tag{5}$$

Now, each keyterm is comparable in the whole document. The score of a candidate noun phrase $p_i \in P$ is the sum of scores of keyterms contained in it, as (6) shows.

$$\mathrm{Score}\left(p_i\right) = \sum_{kt_j \in p_i} \mathrm{Score}\left(kt_{ij}\right). \tag{6}$$

After we get the scores of all candidate noun phrases, we select the top-K phrases with the largest ranking scores as the keyphrases of the input document.

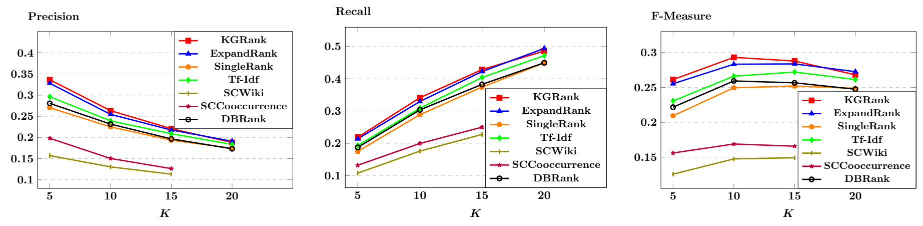
6 Experimental Evaluation

For ease of presentation, our proposed knowledge graph based method to rank keyterms for keyphrase extraction is denoted as *KGRank*.

6.1 Datasets and Evaluation Metrics

We use the dataset DUC2001[3] to evaluate the performance of our method. The manually labeled keyphrases on this dataset are created in the work [27]. The dataset contains 308 news articles. The average length of the documents is about 700 words. And each document is manually assigned about 10 keyphrases. Since our task is to extract noun phrases as defined in Sect. 2.1, to construct the golden standard, we drop adjectives from the manual phrases.

[3] http://www-nlpir.nist.gov/projects/duc/past_duc/duc2001/data.html



(a) Precision vs. K (b) Recall vs. K (c) F-measure vs. K

Fig. 5. Comparison with other algorithms

In the experiments, we adopt precision, recall, and F-measure to evaluate the performance of keyphrase extraction. Their formal definitions are given in (7), (8), and (9) respectively, where $\mathrm{count}_{\mathrm{correct}}$ is the number of correct keyphrases extracted by automatic method, $\mathrm{count}_{\mathrm{output}}$ is the number of all keyphrases extracted automatically, and $\mathrm{count}_{\mathrm{manual}}$ is the number of manual keyphrases. We use Wikipedia and DBpedia as knowledge graphs respectively to assist keyphrase extraction.

The codes to preprocess data are implemented in Python. The other algorithms are implemented in C++. All experiments are conducted on a Windows Server with 2.4GHz Intel Xeon E5-4610 CPU and 384GB memory.

$$\mathrm{precision} = \frac{\mathrm{count}_{\mathrm{correct}}}{\mathrm{count}_{\mathrm{output}}}. \tag{7}$$

$$\mathrm{recall} = \frac{\mathrm{count}_{\mathrm{correct}}}{\mathrm{count}_{\mathrm{manual}}}. \tag{8}$$

$$\mathrm{F-measure} = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}. \tag{9}$$

6.2 Comparison with Other Algorithms

We compare our method *KGRank* (Wikipedia as the knowledge graph) and *DBRank* (DBpedia as the knowledge graph) with two co-occurrence graph based methods SingleRank [27] and ExpandRank [27], one cluster-based method SCCooccurrence [18], one statistic-based clustering method SCWiki [18], and tf-idf based methods. Our method which adopts Wikipedia as the knowledge graph achieves better performance than using DBpedia. The reason we think is due to the better coverage ratio of keyterms in Wikipedia, which is about 0.8, while the coverage ratio of DBpedia is about 0.6. For *KGRank*, the best results are achieved at $r = 3$ and $\tau = 3.0$ (Euclidean distance threshold). For *DBRank*, the best results are achieved at $r = 5$ and $\tau = 0.4$ (cosine similarity threshold). The windowsize W for SingleRank is varied from 2 to 20. The best results are achieved at $W = 17$. For ExpandRank, we set the expanded document number $k = 5, 10$, and vary windowsize W from 2 to 20. The best results are achieved at $k = 10$ and $W = 17$.

Figure 5 shows the performances of the seven methods on precision, recall, and F-measure respectively, where the keyphrase number $K = 5, 10, 15, 20$. It is clear that our method almost always performs the best. With the growth of K, the precision decreases

Table 4. Case study: keyphrase extraction results for the example document

Method	Keyphrases
KGRank	**florida**, **hurricane andrew**, emergency relief cost, **disaster**, relief, persident carter, **president george bush**,president, dollars, **election campagin**
ExpandRank	**hurricane andrew**, american insurance services group, **president george bush**, andrew card, **florida**, us insurer, president carter, federal emergency management agency, emergency relief cost, president
SingleRank	**president george bush**, president carter, **hurricane andrew**, american insurance services group, andrew card, president, white house chief, emergency relief cost, grain export programme, federal emergency management agency
SCWiki	sign, **election campaign**,**president george bush**, initial response, administration, assistance, business, farm, presidential rival
SCCooccurrence	sign, **hurricane andrew**, inhabitant, **florida**, **louisiana**, **election campaign**, **president george bush**, initial response, president, administration
Tf-idf	**president george bush**, andrew card, **hurricane andrew**, american insurance services group, president carter, emergency relief cost, wallace stickney, federal emergency management agency, grain export programme, james baker

and recall increases. F-measure achieves the best performance when K is 10, which is the average number of manual keyphrases. Note that SC selects keyphrases according to the exemplars, it is not able to control the number of keyphrases. For example, it cannot return enough phrases when $K = 20$.

Table 4 shows the keyphrases extracted by six methods above from the example document when $K = 10$. The keyphrases in red and bold are correct keyphrases.

6.3 Effect of Noun Phrase

To validate our opinion that *different annotators reach high agreement on noun phrases*, we randomly select 20 articles and annotate keyphrases that may contain adjectives for each article, denoted by S. Then we compare our manually labeled keyphrases with the benchmark that is used in the work [27], denoted by T. We find that the proportion of common general phrases is $|S \cap T|/|S \cup T|$ = 44%. Meanwhile, the proportion of common noun phrases is $|S' \cap T'|/|S \cup T|$ = 70%, where S' and T' are the keyphrases by removing adjectives from S and T, respectively. Thus, *this observation confirms our conclusion that different annotators reach high agreement on noun phrases.*

The phenomenon above also can be convinced through automatic keyphrase extraction methods. We compare the common keyphrases extracted by different methods, SingleRank [27], ExpandRank [27], and SCCooccurrence [18]. The rate of noun phrases in the common general keyphrases that are extracted by the three methods is 72%.

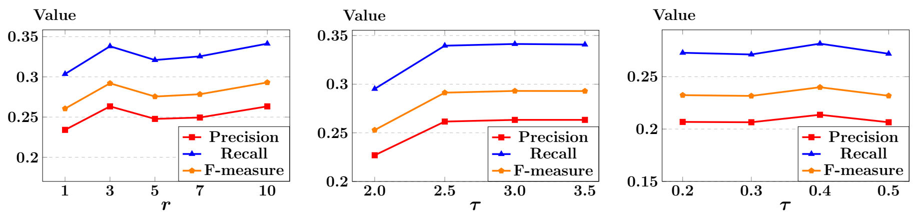
To further study the two types of candidate phrases, we use phrases containing adjectives as keyterms and noun phrases as keyterms respectively to extract keyphrases. The results are shown in Table 5. It is clear that noun phrases perform better than adjective noun phrases.

6.4 Effect of Clustering Parameter

There are a lot of clustering algorithms (e.g., hierarchical clustering, k-means and spectral clustering) and similarity measures (e.g., cosine similarity, Wikipedia-based sim-

Table 5. Adjective noun phrase vs. noun phrase

Phrase Type	Precision	Recall	F-Measure
Adjective Noun Phrase	0.16461	0.215415	0.184411
Noun Phrase	0.205	0.30502	0.243754



(a) Effect of r (Wikipedia) (b) Effect of τ (Wikipedia) (c) Effect of τ (DBpedia)

Fig. 6. Effect of cluster number r and similarity threshold τ

ilarity, and Euclidean distance) can be used in the task of clustering keyterms. Since Google Word2vec performs well in measuring semantic similarity between words and k-means is a simple and effective method to clustering high-dimensional points, we adopt Word2vec vector and k-means in this paper.

We study the impacts of cluster number r in k-means on performance. Figure 6(a) shows the effects of cluster number r when semantic relatedness threshold τ is 3.0 and the keyphrase number K is 10 using Wikipedia as the knowledge graph. As shown in Fig. 6(a), the topic based clustering could improve the performance of keyphrase extraction. Keyterms in the same cluster tend to have more semantic relations, so the h-hop keyterm graph would contain less noises.

6.5 Effect of Keyterm Graph

Due to the large vertex degree of Wikipedia and DBpedia, we fix $h = 2$ in our experiments. If we do not screen out the expanded nodes, the extracted keyterm graph would inevitable contain a lot of nodes that are not related to the topics of the input document. So we use the semantic distance between expanded nodes and anchor nodes to filter the noisy expanded nodes. In this subsection, we study how the semantic relatedness threshold τ affects the performance of keyphrase extraction.

Figure 6(b) and 6(c) show the effects of relatedness threshold τ using Wikipedia and DBpedia respectively, where the cluster number r is 10 and the keyphrase number K is 10. From the two figures, we can see a suitable semantic relatedness threshold could improve the performance since it filters some noisy expanded nodes. A large semantic relatedness threshold would decrease the improvement of performance since it filters too much semantic relations including meaningful relations and expanded nodes.

7 Related Work

The co-occurrence based methods, e.g., *TextRank* [21], *SingleRank* [27], *ExpandRank* [27], build a word co-occurrence graph and apply PageRank or its variation to rank the words. *TextRank* [21] is the first to solve the task by building a word co-occurrence graph and applying PageRank algorithm on the graph to rank words. If top-k ranked words are adjacency in the document, then they combine a phrase. *ExpandRank* [27] explores neighborhood information to help keyphrase extraction. Given a document, it first adopts cosine similarity to find the top-k similar documents and combines the $k+1$ documents as a set. Then it builds a word co-occurrence graph on the document set and gets the ranking score of each word by PageRank. Finally, the score of a candidate phrase is the sum of the scores of the words contained in it. *SW* [11] adopts connectedness centrality, betweenness centrality, and a combination centrality algorithm to rank candidate phrases. Experiments show that there is no obvious difference in the three measures. And they achieve comparable performances with supervised methods KEA [29]. *CM* [5] compares several centrality measures: degree, closeness [2], betweenness, and eigenvector centrality. The experiments on three datasets show that degree is the best measure in the undirected graph, which indicates that tf is a very important feature for keyphrase extraction. So in our method, we choose to use PPR algorithm to rank the keyterms, where the jump probability is set to be proportional to tf.

Instead of just using the input document, the statistic-based methods, e.g., *Wan'07* [28], *SC* [18], *TPR* [17], exploit some external resources. *TPR* [17] builds a word co-occurrence graph. The difference is that it trains Latent Dirichlet Allocation (LDA) model [4] on Wikipedia articles and gets some topics on the words. Then it runs Topical PageRank [10] under each topic, where the jump probability of each word is related with its occurrence probability under the current topic. The final ranking score of each word is the sum of its ranking score under each topic multiply by its occurrence probability given the topic. However, for a single document, the number of topics is much smaller than the topical number used in the experiments.

8 Conclusions

In this paper, we study on automatic single-document keyphrase extraction task. We propose a novel method that combines semantic similarity clustering algorithms with knowledge graph structure to help discover semantic relations hidden in the input document and cover more topics. We also design a measure to compare different clusters by constructing a semantic graph for the centroids of all clusters. Experiments show that our method achieves better performances than the state-of-art methods.

9 Acknowledgements

This work is supported by Research Grant Council of Hong Kong SAR No. 14221716 and The Chinese University of Hong Kong Direct Grant No. 4055048 and NSFC under grant No. 61622201, 61532010, 61370055, 61402020 and Ph.D. Programs Foundation of Ministry of Education of China No. 20130001120021.

References

1. Wikipedia. `http://en.wikipedia.org/`.
2. A. Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.
3. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - A crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009.
4. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
5. F. Boudin. A comparison of centrality measures for graph-based keyphrase extraction. In *IJCNLP 2013*, pages 834–838, 2013.

6. R. Cilibrasi and P. M. B. Vitányi. The google similarity distance. *CoRR*, abs/cs/0412098, 2004.
7. L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.
8. M. P. Grineva, M. N. Grinev, and D. Lizorkin. Extracting key terms from noisy and multi-theme documents. In *WWW 2009*, pages 661–670, 2009.
9. K. M. Hammouda, D. N. Matute, and M. S. Kamel. Corephrase: Keyphrase extraction for document clustering. In *MLDM 2005*, pages 265–274, 2005.
10. T. H. Haveliwala. Topic-sensitive pagerank. In *WWW 2002*, pages 517–526, 2002.
11. C. Huang, Y. Tian, Z. Zhou, C. X. Ling, and T. Huang. Keyphrase extraction using semantic networks structure analysis. In *ICDM 2006*, pages 275–284, 2006.
12. A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP 2003*, pages 216–223, 2003.
13. A. Hulth. Reducing false positives by expert combination in automatic keyword indexing. In *RANLP 2003*, pages 367–376, 2003.
14. G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 538–543, 2002.
15. X. Jiang, Y. Hu, and H. Li. A ranking approach to keyphrase extraction. In *SIGIR 2009*, pages 756–757, 2009.
16. M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML 2015*, pages 957–966, 2015.
17. Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *EMNLP*, pages 366–376, 2010.
18. Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *EMNLP 2009*, pages 257–266, 2009.
19. S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Information Theory*, 28(2):129–136, 1982.
20. C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL 2014*, pages 55–60, 2014.
21. R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *EMNLP 2004*, pages 404–411, 2004.
22. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, 2013.
23. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.
24. S. J. Russell and P. Norvig. *Artificial intelligence - a modern approach: the intelligent agent book*. Prentice Hall series in artificial intelligence. Prentice Hall, 1995.
25. G. Tsatsaronis, I. Varlamis, and K. Nørvåg. Semanticrank: Ranking keywords and sentences using semantic graphs. In *COLING 2010*, pages 1074–1082, 2010.
26. P. D. Turney. Learning to extract keyphrases from text. *CoRR*, cs.LG/0212013, 2002.
27. X. Wan and J. Xiao. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Trans. Inf. Syst.*, 28(2), 2010.
28. X. Wan, J. Yang, and J. Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *ACL 2007*, 2007.
29. I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries*, pages 254–255, 1999.
30. E. Youn and M. K. Jeong. Class dependent feature scaling method using naive bayes classifier for text datamining. *Pattern Recognition Letters*, 30(5):477–485, 2009.