

# Keyphrase Extraction Using Knowledge Graphs

Wei Shi<sup>1</sup>  · Weiguo Zheng<sup>1</sup> · Jeffrey Xu Yu<sup>1</sup> · Hong Cheng<sup>1</sup> · Lei Zou<sup>2</sup>

Received: 11 October 2017 / Revised: 27 October 2017 / Accepted: 5 November 2017 / Published online: 16 November 2017  
© The Author(s) 2017. This article is an open access publication

**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-the-art methods including the graph-based co-occurrence methods and statistic-based clustering methods.

**Keywords** Keyphrase extraction · Knowledge graph · Graph-based algorithms

## 1 Introduction

The continuously increasing text data, such as news, articles, and scientific 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 [8], text classification [35], and document summarization [32]. 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 [11, 12, 14, 30, 34], 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 [26]. 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* [10], *TextRank* [22], *ExpandRank* [31], *CM* [3], build a word co-occurrence graph exploiting the word co-occurrence relations that are obtained

---

✉ Wei Shi  
shiw@se.cuhk.edu.hk  
Weiguo Zheng  
wgzheng@se.cuhk.edu.hk  
Jeffrey Xu Yu  
yu@se.cuhk.edu.hk  
Hong Cheng  
hcheng@se.cuhk.edu.hk  
Lei Zou  
zoulei@pku.edu.cn

<sup>1</sup> The Chinese University of Hong Kong, Shatin, Hong Kong, China

<sup>2</sup> Peking University, Beijing, China

**Table 1** Overview of unsupervised keyphrase extraction approaches

Category	Methods	Information Sources
Co-occurrence based	CM [3], SW [10], TextRank [22], ExpandRank [31]	Input documents
Statistic-based	Wan'07 [32], SC [17], TPR [16]	External text corpus, e.g., Wikipedia article
Semantics-based	Our method	Knowledge graphs, e.g., DBpedia

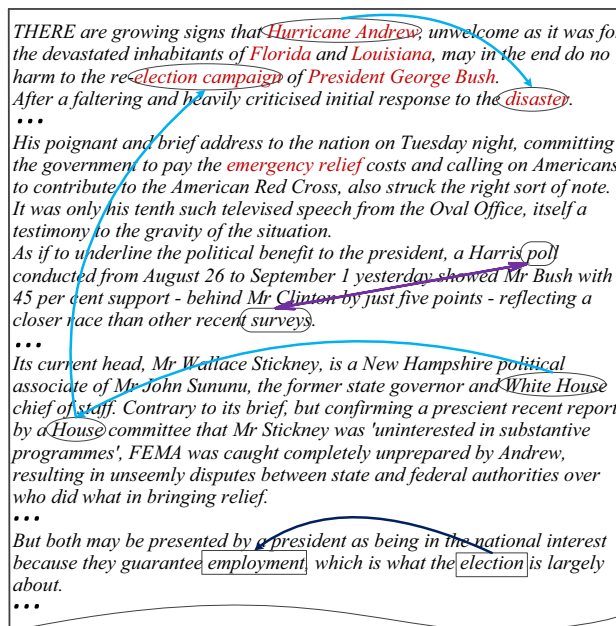
from the input document, and then apply some ranking algorithms such as PageRank [24] and betweenness [6] on the graph to get the ranking score of each word. Based on the ranking score, the *top-K* phrases are returned as keyphrases. The statistic-based methods, e.g., Wan'07 [32], SC [17], and TPR [16], explore some external text corpus to assist keyphrase extraction. Generally, they obtain external information from the statistical information in large corpus. For example, SC [17] uses the Wikipedia<sup>1</sup> 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) [5]. Then these statistical distances are used as the similarity metrics in term clustering and candidate phrases contain cluster exemplar terms are selected as keyphrases. TPR resorts to Wikipedia articles to train a topic distribution for terms to assist keyphrase extraction.

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 [10], 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 (see Sect. 7.1), 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

**Fig. 1** An example of input document

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.

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. 3.1). As shown in the example document in Fig. 1, although the distance between “disaster” and “Hurricane Andrew” in the document is

<sup>1</sup> <http://en.wikipedia.org/>.

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 [1], captures lots of entities, and describes the relationships between the entities. Note that two earlier work communityRank [7] and SemanticRank [28] 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 clustering and graph ranking for keyphrase extraction.
- Extensive experiments over real data show that our method achieves better performance compared with the classic co-occurrence-based methods and statistic-based methods.

This paper is an extension work of our conference paper [27] and improves the existing work by: (1) We combine all the extracted keyterm graphs into an integrated graph and then apply Personalized PageRank (PPR) [9] on it. While in the previous work, PPR is applied on each keyterm graph. (2) We apply three different clustering algorithms in this work and analyze the effect of clustering methods on the performance of keyphrase extraction. (3) We incorporate the importance of different semantic relations (predicates in the knowledge graph) into the extracted keyterm graphs by assigning different weights to the edges. While in the previous work, we ignore the significance of different semantic relations and build unweighted keyterm graphs. (4) We also take two new features of phrases, phrase frequency and first occurrence position, into consideration to improve the performance of keyphrase extraction.

The rest of this paper is organized as follows. In Sect. 2, we review related works in recent years. Section 3 formulates the task of the work and gives an overview of our

method. To cover more topics of a document, we perform several clustering algorithms based on semantics metrics in Sect. 4. Section 5 studies how to incorporate knowledge graphs to model the keyterms, followed by the process of generating keyphrases in Sect. 6. Section 7 reports experiments on real data. Finally, we conclude this work in Sect. 8.

## 2 Related Work

*Entity Recognition and Linking* Given a piece of text and a knowledge base, the task of entity recognition aims to recognize the boundary of entity mentions in the text, and the task of entity linking is to map the entity mentions to the corresponding entities in the knowledge base. Usually, entity recognition is carried out before entity linking. In the literature, there are two typical works, Wikify [21] and DBpedia Spotlight [20], sequentially combine these two tasks together to annotate a piece of text with Wikipedia and DBpedia, respectively. Wikify emphasizes on the annotation of important phrases in a document with Wikipedia. It first builds a controlled vocabulary composed of Wikipedia article titles and surface forms which occur at least 5 times, next selects keywords or keyphrases from all the n-grams which occur in the controlled vocabulary using a very simple unsupervised keyphrase extraction method as the recognized entity mentions, and then adopts a word sense disambiguation method mainly based on context similarity to find the corresponding Wikipedia article for each entity mention. DBpedia Spotlight highlights the annotation of a document with DBpedia under a user defined environment. DBpedia provides several choices such as the annotated entity type, annotation confidence to users to control the whole entity recognition and linking process. Typically, DBpedia Spotlight uses a LingPipe Exact Dictionary-based Chunker<sup>2</sup> to obtain the entity mentions. For entity linking, it regards all the paragraphs containing the anchor link (e.g., a pair of entity mention and entity, which occurs as an anchor link in the Wikipedia article) in Wikipedia articles as the context of the entity and the paragraphs containing the entity mention in the document as the context of the mention. Then it adopts vector space model to represent contexts and uses cosine similarity to measure the similarity between two contexts. The innovation of the work is in the vector space representation, that it measures the importance of a word by its inverse entity frequency instead of the usually inverse document frequency.

To some degree, from the aspect of entity disambiguation, entity recognition and linking is similar with another task, entity set expansion. Formally, entity set expansion is

<sup>2</sup> Alias-i. LingPipe 4.0.0. <http://alias-i.com/lingpipe>, retrieved on 24.08.2010, 2008.

referred as finding a set of related entities given one or several seed entities, such that the entities and the seeds fall into the same category [33]. We investigate two recent works here [4, 25]. [4] focuses on recognizing the long-tail entities which belong to the same semantic class with the given seed entities. Long-tail entities usually have low frequency, which is the reason why most frequency-based methods fail. [4] makes use of the webpage's structural and textual information to train a semi-supervised conditional random field [18] to build a high-quality page-specific extractor. The difference between webpage entity discovery and entity recognition in text data is that the former could resort to the html tag to recognize the candidate entities, while the latter always needs a string segmentation algorithm and inevitably is more difficult and results in more errors. [25] addresses the issue of multi-faceted senses existed in a set of entities. Note that the entity in [25] must appear as a single word. [25] first represents each word as a vector of skip-grams within a predefined window around the word. Next it builds an ego-network (ego equals to seed) which is a word graph with edges added if the Jaccard similarity between the two words is above a threshold. The ego-network naturally forms into communities. Then it combines each community with the most similar Wikipedia lists to form the candidate entities. Finally it uses several heuristic rules to get the expanded entity set. Entity set expansion reduces the disambiguation through the semantic coherence among the whole expanded entity set and do not need to explicitly indicate the concept that the entity set conveying.

**Graph-Based Keyphrase Extraction** The co-occurrence-based methods, e.g., *TextRank* [22], *SingleRank* [31], *ExpandRank* [31], build a word co-occurrence graph and apply PageRank or its variation to rank the words. *TextRank* [22] 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* [31] 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 keyphrase is the sum of the scores of the words contained in the phrase. *SW* [10] adopts connectedness centrality, betweenness centrality, and a combination centrality algorithms 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 [34] and Summarizer [29]. *CM* [3] compares several centrality measures: degree, closeness, 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* [32], *SC* [17], *TPR* [16], exploit some external resources. *Wan'07* [32] combines sentence extraction and keyphrase extraction together based on the point that important sentences and keyphrases reinforce each other. It builds three graphs, which are sentence-to-sentence graph, word-to-word graph and sentence-to-word graph. And the relations in word-to-word graph are established according to the mutual information between words using a large external corpus. *SC* [17] adopts a clustering-based method which aims at achieving good coverage of topics. Terms are single words after stopwords removed. It first computes the similarity between two terms and then clusters all the terms based on similarity measures. Terms which are the cluster centroids are called exemplar terms. Then it selects all the candidate phrases which contain exemplar terms as the keyphrases. *TPR* [16] also builds a word co-occurrence graph. The difference is that it trains Latent Dirichlet Allocation (LDA) model [2] on Wikipedia articles and get some topics on the words. Then it runs Topical PageRank on each topic, where the jump probability 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.

### 3 Problem Definition and 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.

#### 3.1 Problem Formulation

**[Named Entity]** A named entity is a real-world object such as location, organization, and person. "George Bush", "Florida", "Red Cross", "Gulf of Mexico", to name a few, all are named entities.

**[Keyterm]** *Noun words* and *named entities* in the document are called keyterms. For example, "hurricane", "florida", "george bush" are keyterms.

**[Keyphrase]** We define keyphrase as a continuous sequence of keyterms, which is highly important and

**Table 2** Notations

Notation	Definition
$d$	Document
$G = (V, E, LA)$	Knowledge graph
$P = \{p_1, p_2, \dots, p_n\}$	A group of noun phrases
$KT = \{kt_1, kt_2, \dots, kt_m\}$	A group of keyterms
$r$	Number of clusters of $KT$
$AN = \{v_{a_1}, v_{a_2}, \dots, v_{a_r}\}$	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 $AN$
$\tau$	The semantic relatedness threshold

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* Although majority of the existing keyphrase extraction works consider both nouns and adjectives, in this paper, we only consider phrases composed of continuous nouns or named entities. And such phrase is called “noun phrase” in this work, for example, “president George Bush”, “Hurricane Andrew”, “New York City”. The reasons we use noun phrases 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% [31]. (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. 7.4).

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

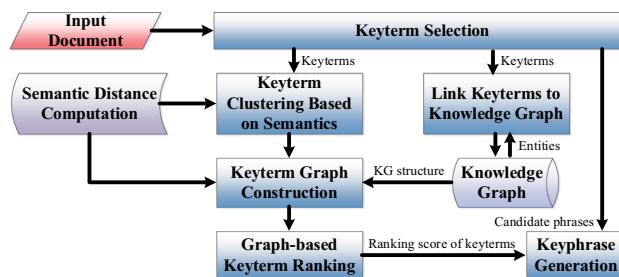
**Table 3** Phrases with and without adjectives

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

### 3.2 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 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 keyterm-keyterm relations in the input document through extracting the  $h$ -hop keyterm graph (please refer to Definition 1) from the knowledge graph. Next, we combine all the extracted keyterm graphs corresponding to all the clusters into an integrated document graph and apply Personalized PageRank [9] on it to obtain the ranking score of each keyterm. The PPR score of a candidate phrase is the sum of the PPR score of the keyterms in it. Then we combine another two phrase features with the phrase PPR score to get the final phrase score and select the  $top-K$  ranking phrases as the keyphrases.

*Keyterm Clustering Based on Semantics* In general, a document consists of multiple topics. Thus we propose to adopt clustering algorithms to divide keyterms into several clusters that keyterms in the same cluster belong to the same topic and different clusters belong to different topics. Specifically, the keyterms are clustered according to the semantic similarity. Then we try to generate keyphrases



**Fig. 2** System framework



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 graph ranking algorithms, such as PPR and SimRank [13], to compute the importance of each keyterm. Thus we get the PPR score of each candidate phrase based on the keyterm score. Next we add two other phrase features to compute the final phrase score. Then the *top-K* candidate phrases with the largest scores are delivered as keyphrases.

## 4 Keyterm Clustering Based on Semantics

Generally, a document covers more than one topics. A topic can be represented by the words contained in it. Moreover, the words in the same topic are very related. Clustering is the task of grouping objects into clusters such that objects in the same cluster are more related or similar. Through clustering on the keyterms in a document, we can divide the document into several groups in which keyterms convey a topic, thus we can get a good topic coverage of the document.

### 4.1 Keyterm Selection

The existing method *SC* [17] 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. Since the knowledge graph consists of entities and relations, it is easy to integrate the knowledge graph into keyphrase extraction by linking keyterms to entities. To extract the keyterms, we resort to the existing NLP tools, e.g., Stanford CoreNLP [19].

### 4.2 Keyterm Clustering

In order to convey the semantics, we use Google Word2vec [23] 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 [15] does. In this paper, we adopt three widely used clustering algorithms, i.e., K-Means, spectral clustering and affinity propagation, to group keyterms in a document based on the semantic relatedness among them.

#### 4.2.1 K-Means

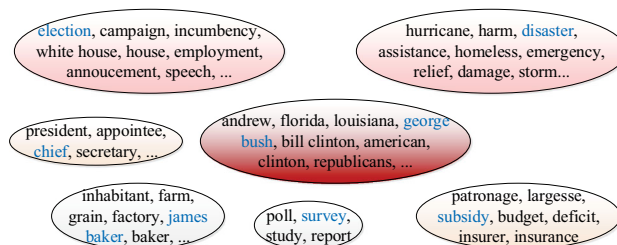
K-Means is one of the simplest unsupervised learning algorithms to solve the well-known clustering problem. It partitions *m* objects into *r* clusters such that each object belongs to the closest cluster in terms of mean distance.

The main steps of K-Means algorithm are as follows: (1) Randomly selects *r* initial clustering centroids. (2) Compute the distance between each object and each centroid, and assign each object to the cluster that has the closest centroid. (3) Recalculate the centroids of *r* clusters after all objects have been assigned. (4) Repeat steps 2 and 3 until the centroids no longer change.

**Example 1** Figure 3 shows the clustering result by K-Means algorithm 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.

#### 4.2.2 Spectral Clustering

Different from K-Means that requires objects to be clustered are points in *N* dimensional Euclidean space, spectral clustering could work for objects not in Euclidean space since it only needs the similarity matrix of the objects. Generally, given *n* objects to be clustered, the process of spectral clustering can be summarized as follows: (1) Construct the similarity matrix of *n* objects, denoted as *A*, where *A* is a  $n \times n$  matrix. (2) Construct the Laplacian matrix *L* based on *A*, where  $L = D - A$  and *D* is the diagonal matrix of *A*. (3) Compute the *top-K* largest eigenvalues and eigenvectors of *L*. (4) Form a  $n \times k$  matrix *A'*, where each column is an eigenvector and columns are ranked in descending order of their corresponding eigenvalues. Each row can be regarded as a newly formed object, which is a *k* dimensional vector. (5) Apply K-Means algorithm on *A'*. The clustering result of each row is the result of the original *n* objects.



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

In our work, the input are keyterm vectors. We compute the similarity matrix by applying a heat kernel on the Euclidean distance matrix of keyterm vectors, as Eq. 1 shows, where  $ED$  is the Euclidean distance matrix,  $std(ED)$  is the global standard deviation of  $ED$ .

$$A_{ij} = e^{-ED_{ij}/std(ED)} \tag{1}$$

In this paper, we adopt the spectral clustering implemented in Python module sklearn.

### 4.2.3 Affinity Propagation

Another clustering algorithm we adopt is affinity propagation, which is based on the message passage between data points. Different from K-Means and spectral clustering, affinity propagation does not need to specify the number of clusters before running the process of clustering.

We give an overview of the process of affinity propagation. First, affinity propagation defines a similarity matrix. Then it iteratively updates two matrices. One of the two matrices is called the responsibility matrix and the other is called the availability matrix. The former is denoted as  $R$ . An element  $r(i, k)$  of  $R$  describes the degree that object  $k$  is suitable as the exemplar of object  $i$  comparing to other candidate exemplars for  $i$ . The latter is denoted as  $A$ . An element  $a(i, k)$  of  $A$  characterizes how well suited that object  $i$  takes object  $k$  as the exemplar. The iteration stops until the cluster boundaries remain unchanged or the predefined number of iterations ran out.

In this work, we use the affinity propagation provided by Python module sklearn. And all the parameters are set as default.

## 5 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.

### 5.1 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 obtain the probability of each candidate entity  $en_i$  by calculating the proportion of its co-occurrence number with  $kt$  over the total occurrence number of  $kt$  as shown in Eq. 2.

$$Pr\{(kt, en_i)|kt\} = \frac{T(kt, en_i)}{T(kt)}, \tag{2}$$

where  $T(kt, en_i)$  is the co-occurrence number of  $kt$  and  $en_i$ , and  $T(kt)$  is the occurrence number of  $kt$ . Given a keyterm  $kt$ , we explore the dictionary to find its candidate mappings. 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. In our work, we use DBpedia as the knowledge graph. To provide the context of entities, we resort to the dataset “long\_abstracts\_en.ttl”.

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

### 5.2 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.,  $AN = \{v_{a_1}, v_{a_2}, \dots, v_{a_i}\} \subset V$ . Definition 1 describes the  $h$ -hop keyterm graph.

**Definition 1** (*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  $i \neq j$ .

same window, an edge is added between  $v_{a_i}$  and  $v_{a_j}$ . For each cluster, we build a keyterm graph. Then we combine all the keyterm graphs together and obtain the keyterm

---

### Algorithm 1 $H$ -hop Keyterm Graph Construction

---

**Input:** Knowledge graph  $G = (V, E, LA)$ , anchor nodes  $AN$ , semantic relatedness threshold  $\tau$

**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  $Path(v_{a_i}, v_{a_j})$ 

```

---

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  $v_{a_j}$  ( $\forall v_{a_i} \in AN$  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  $\tau$ ,  $L$  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  $v_{a_j}$ . If their corresponding keyterms  $kt_i$  and  $kt_j$  occur in the

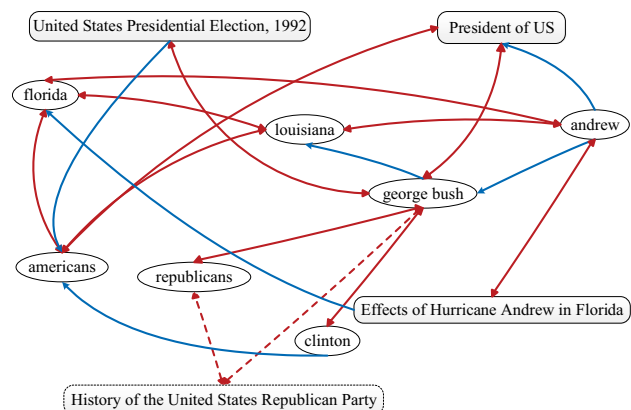
graph for the input document, which is the so-called *document keyterm graph*.

## 6 Keyphrase Generation

According to the  $h$ -hop keyterm graph built in Sect. 5.2, we can generate keyphrases. First, ranking scores of keyterms are calculated, then based on which ranking scores of candidate phrases are computed.

### 6.1 Keyterm Ranking

After constructing the document keyterm graph, the next focus is how to measure the importance of different keyterms. Graph centrality algorithms are designed to measure the importance of nodes within a graph. Various graph centrality algorithms including degree, closeness, betweenness, eigenvector centrality, and PageRank have been adopted in the existing co-occurrence word graph ranking methods.



**Fig. 4**  $H$ -hop keyterm graph



In the co-occurrence based graph (i.e., the keyterm graph is built according to the co-occurrence relations of words), words with larger *tf* (term frequency) tend to have larger vertex degrees. In an undirected graph, PageRank score is closely related to vertex degree. So PageRank score on co-occurrence based graph reflects the *tf* of a word.

However, our keyterm graph is a semantic graph, where edges reflect semantic relations between keyterms. Meanwhile, 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, 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:

$$PPR(v_i) = \lambda \times \sum_{v_j \rightarrow v_i} \frac{PPR(v_j)}{Outdegree(v_j)} + (1 - \lambda) \times p(v_i), \tag{3}$$

where  $PPR(v_i)$  is the PPR score of  $v_i$ ,  $outdegree(v_j)$  is the outdegree of  $v_j$ ,  $p(v_i)$  is the jump probability of  $v_i$  and  $\lambda$  is a damping factor which is usually set to 0.85 [24]. The jump probability  $p(v_i)$  is computed according to Eq. 4, where  $N$  is the number of nodes,  $tf(v_i)$  is the term frequency of  $v_i$ .

$$p(v_i) = \frac{tf(v_i)}{\sum_{i=1}^N tf(v_i)}, \quad i = 1, \dots, N. \tag{4}$$

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 document keyterm graph. Then we obtain the ranking scores of the corresponding anchor nodes.

Note that the edges of the document keyterm graph we built in Sect. 5.2 are with labels, which are called predicates in the knowledge graph. Generally, different predicates have different semantic meanings and play different role in the context. However, we do not distinguish different predicates and regard them as the same in Eq. 3, that is the transition probability from  $v_j$  to  $v_i$  (there is an edge between  $v_j$  between  $v_i$ ) only depends on the outdegree of  $v_j$ . In order to take use of the edge labels, we resort to the idea of *idf* and assign different weight to distinct predicates, as Eq. 5 shows:

$$W(e_i) = \log \frac{|E|}{N(e_i)}, \tag{5}$$

where  $|E|$  is the number of edges in knowledge graph  $G$ ,  $N(e_i)$  is the occurrences of edge  $e_i$ . Then the transition probability between  $v_j$  and  $v_i$  is defined as:

$$M_{ji} = \frac{W(e(v_j, v_i))}{\sum_{v_j \rightarrow v_k} W(e(v_j, v_k))}. \tag{6}$$

And we formulate the PPR on weighted graph as:

$$PPR(v_i) = \lambda \times \sum_{v_j \rightarrow v_i} M_{ji} \cdot PPR(v_j) + (1 - \lambda) \times p(v_i). \tag{7}$$

### 6.2 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 [17, 31] adopt the syntactic rule (JJ) \* (NN|NNS|NNP|NNPS)+, where “JJ” is an adjective, “NN”, “NNS”, “NNP” and “NNPS” 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. 3.1. Hence, we design a rule by removing the adjectives, i.e., (NN|NNS|NNP|NNPS)+. Following this rule, all the candidate phrases are a chain of continuous noun words except named entities. In the example sentence, candidate phrases “federal emergency management agency” and “patronage backwater” are generated.

The PPR ranking score of a candidate noun phrase  $p_i \in P$  is the sum of scores of keyterms contained in it, as shown in Eq. 8.

$$PPRScore(p_i) = \sum_{kt_{ij} \in p_i} PPR(kt_{ij}). \tag{8}$$

We further embed the frequency and the first occurrence position of phrases, denoted by  $tf(p_i)$  and  $pos(p_i)$ , respectively, into Eq. 8 to improve the quality of phrase scores, as shown in Eq. 9.

$$Score(p_i) = \frac{tf(p_i)}{pos(p_i)} \times PPRScore(p_i). \tag{9}$$

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

### 7 Experimental Evaluation

In the following, our proposed *knowledge graph*-based method to *rank* keyterms for keyphrase extraction is denoted as *KGRank*.

#### 7.1 Datasets and Evaluation Metrics

We use the dataset DUC2001<sup>3</sup> to evaluate the performance of our method. The manually labeled keyphrases on this dataset are created in the work [31]. 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. 3.1, to construct the golden standard, we drop adjectives from the manually labeled phrases.

In the experiments, we adopt precision, recall, and F-measure to evaluate the performance of keyphrase extraction. Their formal definitions are given in Eq. 10, Eq. 11, and Eq. 12, respectively, where  $count_{correct}$  is the number of correct keyphrases extracted by automatic method,  $count_{output}$  is the number of all keyphrases extracted automatically, and  $count_{manual}$  is the number of manual keyphrases. We use DBpedia as the knowledge graph 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.

$$Precision = \frac{count_{correct}}{count_{output}} \tag{10}$$

$$Recall = \frac{count_{correct}}{count_{manual}} \tag{11}$$

$$F\text{-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

#### 7.2 Comparison with Other Algorithms

We compare our method *KGRank* with two co-occurrence graph-based methods *SingleRank* [31] and *ExpandRank* [31], one cluster-based method *SCCooccurrence* [17], one statistic-based clustering method *SCWiki* [17].

<sup>3</sup> [http://www-nlpir.nist.gov/projects/duc/past\\_duc/duc2001/data.html](http://www-nlpir.nist.gov/projects/duc/past_duc/duc2001/data.html).

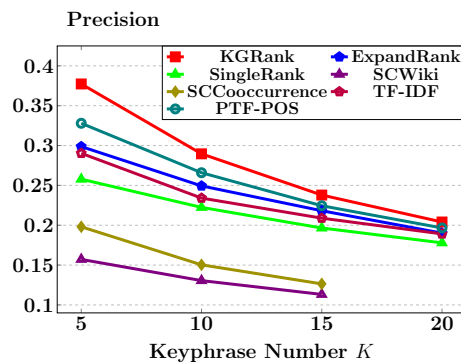


Fig. 5 Compared algorithms: precision

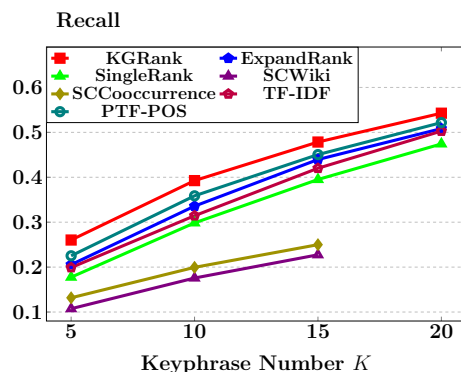


Fig. 6 Compared algorithms: recall

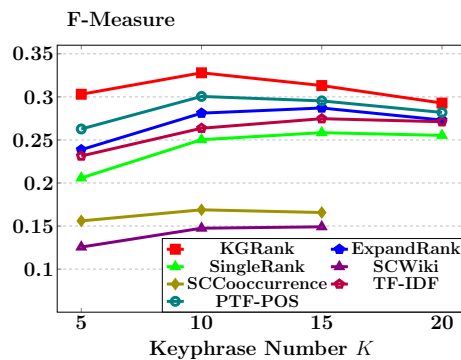


Fig. 7 Compared algorithms: F-measure

Furthermore, we take two very strong baseline methods *TF-IDF* and *PTF-POS* into comparison. And *PTF-POS* uses the same way as ours to combine phrase frequency and the first occurrence phrase position.

Figures 5, 6 and 7 show 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 achieves the best performance, which indicates the effectiveness of the semantic relations obtained from the structure of the knowledge graph in keyphrase extraction. Following closely behind is *PTF-*

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

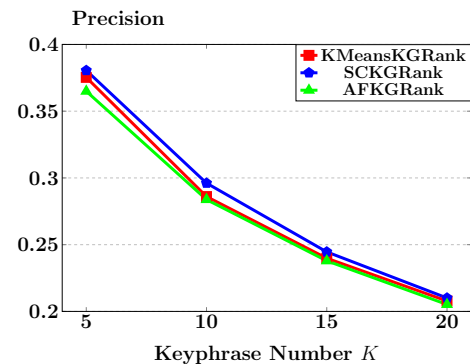
Method	Keyphrases
<i>KGRank</i>	<b>florida, hurricane andrew</b> , emergency relief cost, <b>disaster</b> , relief, president carter, <b>president george bush</b> , president, dollars, <b>election campagin</b>
<i>ExpandRank</i>	<b>hurricane andrew</b> , american insurance services group, <b>president george bush</b> , andrew card, <b>florida</b> , us insurer, president carter, federal emergency management agency, emergency relief cost, president
<i>SingleRank</i>	<b>president george bush</b> , president carter, <b>hurricane andrew</b> , american insurance services group, andrew card, president, white house chief, emergency relief cost, grain export programme, federal emergency management agency
<i>SCWiki</i>	sign, <b>election campaign</b> , <b>president george bush</b> , initial response, administration, assistance, business, farm, presidential rival
<i>SCCooccurrence</i>	sign, <b>hurricane andrew</b> , inhabitant, <b>florida, louisiana, election campaign, president george bush</b> , initial response, president, administration
<i>TF-IDF</i>	<b>president george bush</b> , andrew card, <b>hurricane andrew</b> , american insurance services group, president carter, emergency relief cost, wallace stickney, federal emergency management agency, grain export programme, james baker
<i>PTF-POS</i>	<b>hurricane andrew</b> , president, <b>president george bush, florida</b> , sign, <b>disaster, louisiana</b> , clinton, president announcement, response

POS, which implies phrase frequency and first occurrence position are two very significant features in keyphrase extraction and should not be ignored. Next is ExpandRank, followed by TF-IDF and SingleRank, which indicates that documents with similar topics could provide useful information to aid keyphrase extraction. TF-IDF performs better than SingleRank, which illustrates that distinguishing the importance of different words improves the task performance since SingleRank is very similar with directly using *tf*. At the bottom are SCCooccurrence and SCWiki, which implies that graph-based methods perform better than clustering-based methods on the dataset. SCCooccurrence performs better than SCWiki, which conveys that directly using the statistical information in external corpus without disambiguation may not achieve better performance than using the internal information in the input document. In general, for all the methods, with the growth of  $K$ , the precision decreases 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. Hence, 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 seven methods above from the example document when  $K = 10$ . The keyphrases in red and bold are correct keyphrases.

### 7.3 Effect of Clustering Parameter

In this paper, we adopt three clustering algorithms: K-Means, spectral clustering and affinity propagation. In this section, we evaluate the effect of different clustering

**Fig. 8** Clustering methods: precision

methods on the performance of keyphrase extraction. While K-Means and spectral clustering need to predefine the number of clusters, affinity propagation would set the number of clusters automatically. Thus in order to make comparison, we set the number of clusters in K-Means and spectral clustering the same with that of affinity propagation. And their performances on keyphrase extraction are shown in Figs. 8, 9 and 10, where keyphrase number  $K$  is set as 5, 10, 15 and 20. KMeansKGRank, SCKGRank, and AFKGRank are our methods which adopt K-Means, spectral clustering and affinity propagation as the corresponding clustering algorithm, respectively. From these figures, we can see that spectral clustering beats the other two algorithms. The reason lies in that there are duplicates in the representation of keyterms as a point in the high-dimensional space.

Since K-Means and spectral clustering need to indicate the number of clusters, we also study the effect of cluster

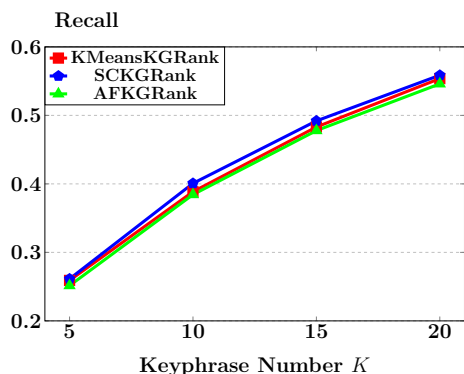


Fig. 9 Clustering methods: recall

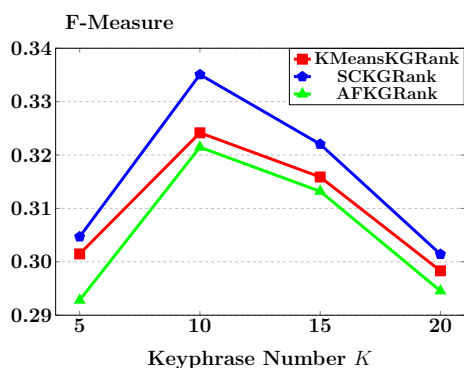


Fig. 10 Clustering methods: F-measure

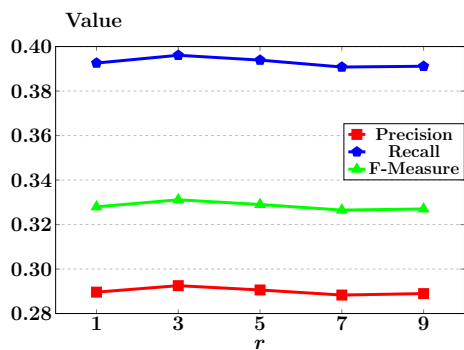


Fig. 11 Effect of  $r$  (K-Means)

number  $r$  in these two algorithms on keyphrase extraction. Figures 11 and 12 show the effects of cluster number  $r$  on the task when keyphrase number  $K$  is 10. As demonstrated in these two figures, keyterm clustering could improve the performance of keyphrase extraction. The reason is that keyterms in the same cluster tend to be more related to each other, so the  $h$ -hop keyterm graph would contain less noises.

### 7.4 Effect of Noun Phrase

To validate our opinion *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 [31], 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 [31], ExpandRank [31], and SCCooccurrence [17]. 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.

### 7.5 Effect of Keyterm Graph

Due to the large vertex degree of DBpedia, we fix  $h = 2$  (the length of extracted paths) in our experiments. If we do not screen out the expanded nodes, the extracted keyterm graph would 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  $\tau$  affects the performance of keyphrase extraction.

We compute the semantic relatedness between each expanded node and the anchor node set, denoted as  $\mu(EV.L, AN)$  in Sect. 5.2. Then we rank all the expanded nodes in descending order of their semantic relatedness scores. We filter the less related expanded nodes given a relatedness threshold  $\tau$ . We introduce another parameter  $\alpha$  to easily represent  $\tau$ . We set  $\tau$  based on  $\alpha$  (a smaller  $\alpha$  results in a larger  $\tau$ ). For each document, we set the number of expanded nodes no larger than the number of anchor nodes. Figure 13 shows the effects of  $\alpha$  on keyphrase extraction, where the cluster number  $r$  is 1 and the keyphrase number  $K$  is 10. From the figure, 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

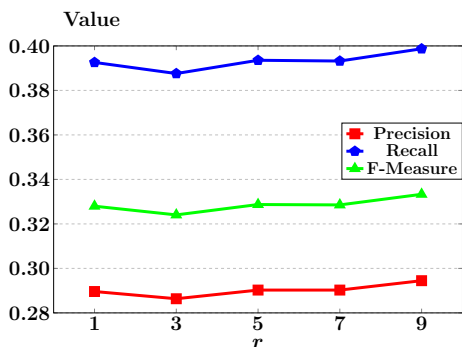


Fig. 12 Effect of  $r$  (spectral clustering)

Table 5 Adjective noun phrase versus 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

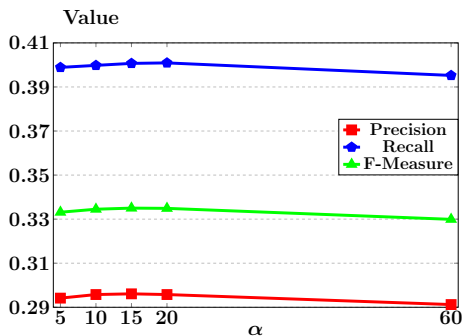


Fig. 13 Effect of semantic relatedness

improvement of performance since it filters too much semantic relations including meaningful relations and expanded nodes.

### 7.6 Effect of Edge Weight

Generally, transition probability in Personalized PageRank makes a difference on the steady state of the random walk. In our method, transition probability depends on the edge weight. So we study the effect of edge weight on the performance of keyphrase extraction in this part. We assign edge weight as shown in Eq. 5, which is inspired from the thoughts of *idf*. We compare it with *KGRank*, which exploits the unweighted document keyterm graph on precision, recall and F-measure. Figures 14, 15 and 16 show the performances when keyphrase number  $K = 5, 10, 15, 20$ . From the figures, we can see that suitable edge weights could improve the performance.

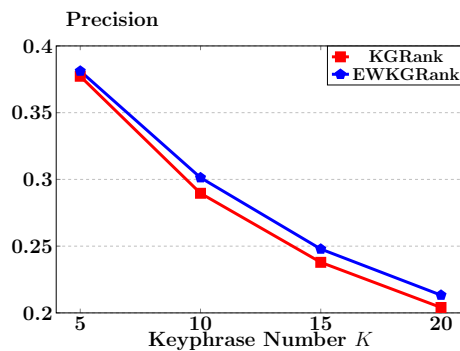


Fig. 14 Effect of edge weight: precision

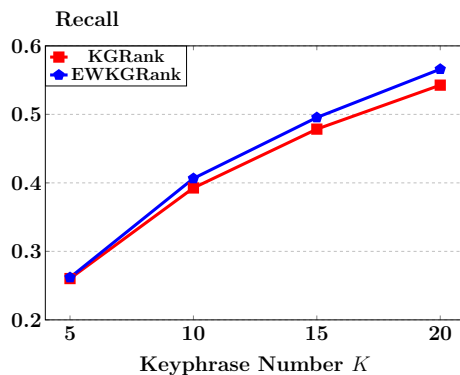


Fig. 15 Effect of edge weight: recall

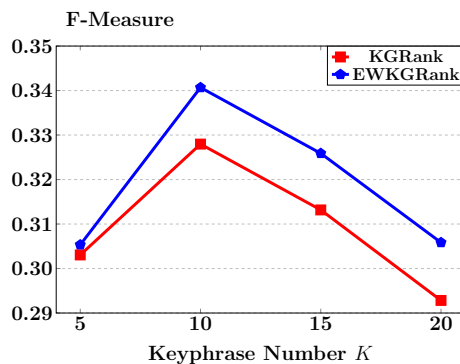


Fig. 16 Effect of edge weight: F-measure

## 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. By integrating the semantic document keyterm graph extracted from the knowledge graph with two widely used phrase features, phrase frequency and the first occurrence position, our



method improves the performance of keyphrase extraction a lot. Experiments show that our method achieves better performances than the state-of-the-art methods.

**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 Nos. 61622201, 61532010, 61370055, 61402020 and Ph.D. Programs Foundation of Ministry of Education of China No. 20130001120021.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S (2009) Dbpedia—a crystallization point for the web of data. *J Web Sem* 7(3):154–165
- Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. *J Mach Learn Res* 3:993–1022
- Boudin F (2013) A comparison of centrality measures for graph-based keyphrase extraction. In: *IJCNLP 2013*, pp 834–838
- Chen Z, Cafarella MJ, Jagadish HV (2016) Long-tail vocabulary dictionary extraction from the web. In: *Proceedings of the ninth ACM international conference on Web search and data mining*, San Francisco, CA, USA, February 22–25, 2016, pp 625–634
- Cilibrasi R, Vitányi PMB (2004) The google similarity distance. *CoRR*, abs/cs/0412098
- Freeman LC (1977) A set of measures of centrality based on betweenness. *Sociometry* 40:35–41
- Grineva MP, Grinev MN, Lizorkin D (2009) Extracting key terms from noisy and multitheme documents. In: *WWW 2009*, pp 661–670
- Hammouda KM, Matute DN, Kamel MS (2005) Corephrase: keyphrase extraction for document clustering. In: *MLDM 2005*, pp 265–274
- Haveliwala TH (2002) Topic-sensitive pagerank. In: *WWW 2002*, pp 517–526
- Huang C, Tian Y, Zhou Z, Ling CX, Huang T (2006) Keyphrase extraction using semantic networks structure analysis. In: *ICDM 2006*, pp 275–284
- Hulth A (2003) Improved automatic keyword extraction given more linguistic knowledge. In: *EMNLP 2003*, pp 216–223
- Hulth A (2003) Reducing false positives by expert combination in automatic keyword indexing. In: *RANLP 2003*, pp 367–376
- Jeh G, Widom J (2002) 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, pp 538–543
- Jiang X, Hu Y, Li H (2009) A ranking approach to keyphrase extraction. In: *SIGIR 2009*, pp 756–757
- Kusner MJ, Sun Y, Kolkin NI, Weinberger KQ (2015) From word embeddings to document distances. In: *ICML 2015*, pp 957–966
- Liu Z, Huang W, Zheng Y, Sun M (2010) Automatic keyphrase extraction via topic decomposition. In: *EMNLP*, pp 366–376
- Liu Z, Li P, Zheng Y, Sun M (2009) Clustering to find exemplar terms for keyphrase extraction. In: *EMNLP 2009*, pp 257–266
- Mann GS, McCallum A (2010) Generalized expectation criteria for semi-supervised learning with weakly labeled data. *J Mach Learn Res* 11:955–984
- Manning CD, Surdeanu M, Bauer J, Finkel JR, Bethard S, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: *ACL 2014*, pp 55–60
- Mendes PN, Jakob M, García-Silva A, Bizer C (2011) Dbpedia spotlight: shedding light on the web of documents. In: *I-SEMANTICS*, pp 1–8
- Mihalcea R, Csomai A (2007) Wikify!: linking documents to encyclopedic knowledge. In: *CIKM 2007*, pp 233–242
- Mihalcea R, Tarau P (2004) Textrank: bringing order into text. In: *EMNLP 2004*, pp 404–411
- Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. *CoRR*
- Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web
- Rong X, Chen Z, Mei Q, Adar E (2016) Egocet: exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In: *Proceedings of the ninth ACM international conference on Web search and data mining*, San Francisco, CA, USA, February 22–25, 2016, pp 645–654
- Russell SJ, Norvig P (2003) *Artificial intelligence—a modern approach: the intelligent agent book*. Prentice Hall
- Shi W, Zheng W, Yu JX, Cheng H, Zou L (2017) Keyphrase extraction using knowledge graphs. In: *Web and Big Data—first international joint conference, APWeb-WAIM 2017, Beijing, China, July 7–9, 2017, Proceedings, Part I*, pp 132–148
- Tsatsaronis G, Varlamis I, Nørvgå K (2010) Semanticrank: ranking keywords and sentences using semantic graphs. In: *COLING 2010*, pp 1074–1082
- Turney PD (2002) Learning algorithms for keyphrase extraction. *CoRR*, cs.LG/0212020
- Turney PD (2002) Learning to extract keyphrases from text. *CoRR*, cs.LG/0212013
- Wan X, Xiao J (2010) Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Trans Inf Syst* 28(2):8. <https://doi.org/10.1145/1740592.1740596>
- Wan X, Yang J, Xiao J (2007) Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, pp 552–559
- Wang RC, Cohen WW (2007) Language-independent set expansion of named entities using the web. In: *Proceedings of the 7th IEEE international conference on data mining (ICDM 2007)*, October 28–31, 2007, Omaha, Nebraska, USA, pp 342–350
- Witten IH, Paynter GW, Frank E, Gutwin C, Nevill-Manning CG (1999) KEA: practical automatic keyphrase extraction. In: *Proceedings of the fourth ACM conference on digital libraries*, pp 254–255
- Youn E, Jeong MK (2009) Class dependent feature scaling method using naive Bayes classifier for text datamining. *Pattern Recognit Lett* 30(5):477–485