Competitor Mining from Web Encyclopedia: A Graph Embedding Approach

Xin Hong[1,2], Peiquan Jin[1,2(✉)], Lin Mu[1,2], Jie Zhao[3], and Shouhong Wan[1,2]

[1] University of Science and Technology of China, Hefei 230027, Anhui, China
jpq@ustc.edu.cn
[2] Key Laboratory of Electromagnetic Space Information, China Academy of Science, Hefei 230027, Anhui, China
[3] School of Business, Anhui University, Hefei 230601, Anhui, China

Abstract. Mining competitors from the web has been a valuable and emerging topic in big data and business analytics. While normal web pages may include incredible information like fake news, in this paper, we aim to extract competitors from web encyclopedia like Wikipedia and DBpedia, which provide more credible information. We notice that the entities in web encyclopedia can form graph structures. Motivated by this observation, we propose to extract competitors by employing a graph embedding approach. We first present a general framework for mining competitors from web encyclopedia. Then, we propose to mine competitors based on the similarity among graph nodes and further present a similarity computation method combing graph-node similarity and textual relevance. We implement the graph-embedding-based algorithm and compare the proposed method with four existing algorithms on the real data sets crawled from Wikipedia and DBpedia. The results in terms of precision, recall, and F1-measure suggest the effectiveness of our proposal.

Keywords: Web encyclopedia · Competitor mining · Similarity measurement · Heterogeneous graph

1 Introduction

Competitor mining [1] has been an important issue in the big data era. With the advance of the Internet, competition is becoming more serious in almost every business area. Especially in Internet-based businesses, new companies appear frequently, which makes it difficult to detect competition situations in the market. For instance, if one new company plans to start research and development of Internet-based intelligent tutoring, it is much hard for the company to know the competence situation because of the rapid development of the area. Therefore, an automatic intelligent competitor mining tool is needed to provide the newest and complete information about concerned competitors. Meanwhile, the web has become an encyclopedia especially due to the appearance of some web knowledge bases like Wikipedia (www.wikipedia.org) and DBpedia (https://wiki.dbpedia.org/). As a result, mining competitors from web encyclopedia has been a valuable and emerging topic in big data and business analytics.

Z. Huang et al. (Eds.): WISE 2020, LNCS 12342, pp. 56–68, 2020.
https://doi.org/10.1007/978-3-030-62005-9_5

Some existing works are focusing on competitor mining from the Web [1–4]. However, previous studies in competitor mining focused on web pages or entities and neglected the graph-structure information among competitors. In addition. the incredible information like fake news in web pages will make the extracted information untrusted. On the other hand, we found that the rich information in the current web encyclopedia like Wikipedia and DBpedia is helpful to construct a graph structure among competitors, products, business areas, and other information.

In this work, we concentrate on mining competitors from web encyclopedia. Specially, we focus on Wikipedia and DBpedia, as they are among the biggest web encyclopedia in the world. Differing from previous rule-based competitor mining approaches, we consider the graph structure about competitors and propose a new graph embedding approach to mine competitors from web encyclopedia. Briefly, the differences of this study from existing works and the unique contributions of the paper can be summarized as follows:

(1) We propose to mine competitors from web encyclopedia rather than from web pages or search engines. Further, we notice that the entities in web encyclopedia can form intrinsic graph structures. Thus, we propose to mine competitors by combining graph embedding and text mining. To the best of our knowledge, this is the first study that integrates graph embedding and text mining to extract competitors from web encyclopedia.
(2) We develop a framework for mining competitors from web encyclopedia. Particularly, we use the similarity among graph nodes to measure the competitive relationships and propose a similarity computation method based on graph-node similarity and textual relevance. In addition, we give the detailed implementation of the graph embedding approach.
(3) We evaluate the proposed method on data sets crawled from Wikipedia and DBpedia, and compare our algorithm with four existing algorithms. The experimental results suggest the effectiveness of our proposal.

2 Related Work

In this section, we review the related works of this study, including competitor mining, Wikipedia-related studies, and graph embedding.

Competitor Mining. There have been a few similar works studying competitors mining problems based on text mining techniques [1–7]. Bao et al. [1] proposed a novel algorithm called CoMiner, which conducted web-scale mining in a domain-independent method. Bondarenko et al. [2] studied the comparative web search problem in which the user inputted a set of entities and the system returned relevant information about the entities from the web. Chen et al. [5] presented a framework user-oriented text mining. And they discovered knowledge by the form of association rules. Li et al. [6] developed a weakly-supervised bootstrapping method for competitive question identification and comparable entity extraction by adopting a mass online question archive. In addition, Ruan et al. [7] leveraged seeds of competition between companies and competition between products

to distantly supervise the learning process to find text patterns in free texts, i.e., Yahoo! Finance and Wikipedia.

Wikipedia Data Analysis. There are also some previous works focusing on Wikipedia data analysis. Lange et al. [8] presented a system that automatically populates the infoboxes of Wikipedia articles by extracting attribute values from the article's text. This system achieved an average extraction precision of 91% for 1,727 distinct infobox template attributes. Lara Haidar-Ahmad et al. [9] leveraged RDF to build a sentence representation and SPARQL to model patterns and their transformations so that easing the querying of syntactic structures and the reusability of the extracted patterns.

Graph Embedding. Our work is also related to graph mining, which is an important problem in network analysis. Perozzi et al. [10] used local information obtained from truncated random walks to learn latent representation by treating walks as the equivalent of sentences. Grover et al. [11] proposed an algorithm to generalize prior work which was based on rigid notions of network neighborhoods. It defined a flexible notion of a node's network neighborhood and designed a biased random walk procedure. Dong et al. [12] formalized meta-path-based random walks to construct the heterogeneous neighborhoods of a node and then leveraged a heterogeneous skip-gram model to perform node embeddings.

Differing from all previous studies, in this study we propose to integrate graph embedding with textual relevance to improve the effectiveness of competitor mining from web encyclopedia. To the best of our knowledge, this is the first work that considers graph embedding and textual relevance for the competitor-mining problem toward web encyclopedia.

3 Framework for Competitor Mining from Web Encyclopedia

Figure 1 shows the framework of mining competitors from Wikipedia and DBpedia. It consists of several modules, including data crawling, entity extraction, similarity computation, and results ranking.

(1) *Data Crawling*. We craw raw data from Wikipedia and DBpedia. Wikipedia is an online encyclopedia that has a lot of knowledge about entities in the real world. Wikipedia not only contains mass unstructured data but also contains some semi-structured data. Similarly, DBpedia contains much structured information that is made available on the world wide web. Finally, we crawl raw data from Wikipedia and DBpedia using the pattern matching technology [13].
(2) *Entity Extraction.* We mainly extract entities using the infobox information of Wikipedia and some valuable knowledge from DBpedia which contains many kinds of structured knowledge. We utilize the redirection [14] of Wikipedia pages to solve the ambiguity problem between entities. What's more, we also use stem extracting technology to extract the stemming of an industry entity.
(3) *Graph Generating.* We use different node types to create a company knowledge graph. Based on the business background of competitor mining and analysis on some example webpages in Wikipedia, we define four node types, i.e., Industry,

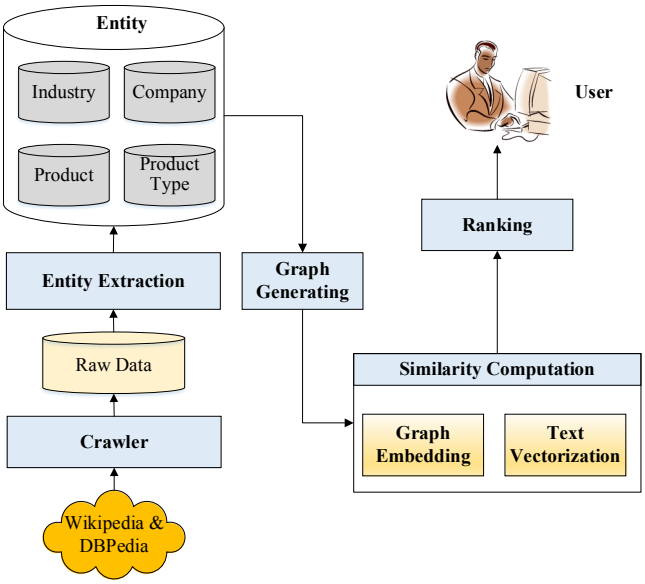


Fig. 1. The framework of competitor mining

Company, Product, Product Type. Industry nodes are regarded as inter-linking nodes between companies. We can also link two companies according to the product information from the companies. There are also some other relationships between entity types, e.g., a company owns an industry and an industry is owned by a company. Consequently, the constructed graph is stored in a Neo4j database.

(4) *Similarity Computation.* The competitive relationships between two companies are mainly determined by their similarity in the real world. Previous works mainly focused on document similarity, but in this paper, we consider graph-structure-based similarity in addition to document similarity. Specially, we first compute graph-node similarity by generating entity vectors according to the graph embedding approach. Further, we employ a heterogeneous graph embedding approach based on a random walk [15] and SkipGram [16] to generate entity vectors. In addition, we compute textual relevance by transforming each entity into a vector using Doc2vec [17]. These two kinds of vectors are finally concatenated and we use the cosine similarity between two concatenated vectors to measure the similarity between two companies.

(5) *Ranking.* The ranking module returns a sorted competitor list for a given company based on the cosine similarity computed by the similarity computation model.

The key point of the proposed framework for competitor mining is that we introduce the graph embedding approach in addition to the textual relevance which has been widely used in text mining. As graphs can describe rich relationships between entities, we can extract competitor relationships by constructing a company knowledge graph derived from web encyclopedia. Based on graph-node similarity and traditional textual

relevance, we can better model the relationships between companies extracted from web encyclopedia.

4 Graph Embedding

In this section, we first introduce some necessary definitions and then formulate the problem. We will use the company as an example to explain the competitive relationship mining problem.

Definition 1 *(Heterogeneous Graph). A heterogeneous graph is denoted as G = (V, E), where V and E represent the set of nodes and edges, respectively. Each node* $v \in V$ *is mapped to a node type using a mapping function* ϕ*, i.e.,* $\phi(v) = r$*, where* $r \in R$ *and R denotes the set of node types in graph G. For a heterogeneous graph G,* $|R|$ *is over 1 as R contains more than one node types.* ■

Definition 2 (*Heterogeneous Edge*). *An edge* $E_{he} = (v_s, v_t) \in E$ *is a heterogeneous edge if two nodes connected by the edge have different node types. We denote* E_{he} *as the set of heterogeneous edges in G.* ■

Definition 3 (*Homogeneous Edge*). *An edge* $E_{ho} = (v_s, v_t) \in E$ *is a homogeneous edge if the two nodes connected by the edge are from the same domain, i.e.,* $\phi(v_s) = \phi(v_t)$*. We use* E_{ho} *to denote the set of homogeneous edges in G.* ■

Definition 4 (*Company Heterogeneous Graph*). *We create a directed label graph of companies G = (V, E,* ϕ*,* φ*) where:*

(1) *V is a set of entities (nodes) in the graph.*
(2) *E is a set of directed edges in the graph representing relationships between entities.*
(3) ϕ*:* $V \rightarrow P(\tau)$ *is a mapping function that assigns each entity v* $V \rightarrow$ *to an entity type. Here,* $\phi(v) \subseteq \tau$*.*
(4) φ*:* $E \rightarrow R$ *is a mapping function that assigns each edge* $e \in E$ *to a relation type. Here,* $\varphi(E) \in R$*.* ■

Based on the above definitions, we can describe the basic idea of the graph-embedding-based approach for competitor mining as follows:

- First, we build a company heterogeneous graph $G = (V, E, \phi, \varphi)$.
- Second, for a given company, we employ a random walk technique to traverse the graph to generate the candidate nodes for the company. These candidate nodes are regarded as competitors candidates.
- Third, we perform a graph-node embedding learning process to learn a vector of company entity embedding from the company heterogeneous graph. We use the learned vector to compute the graph-node similarity between the given company and all the candidate competitors.
- Fourth, we compute the textual relevance between the given company and all the candidate competitors.
- Finally, we combine the graph-node similarity and the textual relevance to rank all candidates. The top-k competitors are returned as the result.

4.1 Random Walk in the Company Heterogeneous Graph

We use random walk technology to traverse the graph to generate candidate nodes, which are considered to contain the relationship between competitors. When we perform a random walk on a homogeneous graph, we select a node by sampling a node from the current node uniformly. In contrast, for a heterogeneous company graph, there are two strategies to choose the next node.

(1) *Skip-Based Random Walk*. This refers to uniformly sampling a node from the nodes connected to v_i through heterogeneous edges. The candidate node set for skipping from v_i to the target node type t is represented by Eq. (1).

$$V_{he}(v_i) = \{v|(v_i, v) \in E_{he} \cup (v, v_i) \in E_{he}\} \tag{1}$$

(2) *Keep-Based Random Walk*. This policy is to uniformly sample one node from those linked to v_i via homogeneous edges. The corresponding candidate set of nodes is denoted by Eq. (2).

$$V_{ho}(v_i) = \{\mathrm{v}|(v_i, v) \in E_{ho} \cup (\mathrm{v}, v_i) \in E_{ho}\} \tag{2}$$

Algorithm 1. Random-walk-based graph construction

Input:
(1) A heterogeneous graph $G = (V,E)$
(2) the initial stay probability α
(3) the number of memorized domains m
(4) the number of random walks per node p
(5) the maximum walk length L_{max}

Output: W, the node-set of random walks.

```
W =∅;
for i = 1 to p do
    for each v∈ V do
        Initialize a random walk by adding v;
        while |W| <L_max do
            Make a Skip or Keep decision according to Eq.(3);
            if Keep then
                Continue W by keeping;
            else if Jump then
                Sample a target node type r according to Eq.(4);
                Continue W by Skipping to node type r;
                Update Q by skipping only last m domains;
            end if
        end while
        Add w to W
    end for
end for
return W
```

In the graph traversal process based on random walks, we aim to uniformly select nodes from a candidate set that mainly includes three aspects. For example, the neighborhood of v_i has only homogeneous edges or only heterogeneous edges. It is also possible to have both homogeneous and heterogeneous edges. Based on the above two choices, we manage the random walk process probabilistically. Then, we calculate the probability by Eq. (3) to decide whether to skip. Here, the parameter of ℓ means that when the node has the same node type as the current node v_i, they will be continuously accessed. If no edge is connected to v_i then we can only link to another node with a different node type. If the neighbor nodes of v_i have no heterogeneous edges, the random walk will be restricted to the same node type. In addition, if there are both heterogeneous and homogeneous edges linked to the current node, we choose the probability $e^{-\alpha\ell}$ to stay in a node with the same node type as the current node. Here, we use an exponential decay function to avoid the long-stay in nodes with the same node type during the random walk process. The next node to jump to is determined by Eq. (4).

$$P(v_i) = \begin{cases} 0 & if\ V_{he}(v_t) = \emptyset \\ 1, & if\ V_{ho}(v_t) = \emptyset \\ e^{-\alpha\ell}, & otherwise \end{cases} \tag{3}$$

$$R_{jump}(v_i) = \begin{cases} \left\{r | r \in T,\ V^r_{Skip}(v_i) \neq \emptyset\right\}, & if\ nonempty \\ \left\{r | r \in S, V^r_{Skip}(v_i) \neq \emptyset\right\}, & otherwise \end{cases} \tag{4}$$

Algorithm 1 shows the details of the random walk with the above-mentioned strategies to sample from the heterogeneous graph [18]. For each node $v \in V$, we initialize a random walk starting from v_i until the maximum length L_{max} is reached.

4.2 Graph-Node Embedding Learning

Based on the number of generated random walk nodes, we use the SkipGram model [16] to train graph node embeddings. The SkipGram model can maximize the co-occurrence probability between the current node and its context nodes. The probability of co-occurrence in the SkipGram model is defined by Eq. (5).

$$P\left(\left(v_i,\ v_j\right) \in T_{walk}\right) = \delta\left(\vec{v}_i \cdot \vec{v}_j\right) \tag{5}$$

Let δ (•) be a sigmoid function, $\vec{v_i}$ and $\vec{v_j}$ be the vectors of v_i and v_j, we can define the negative sampling probability by Eq. (6), where V_F represents the negative sampling node randomly selected. In summary, the model needs to maximize the loss function, as shown in Eq. (7).

$$P((v_i,\ v_F) \notin T_{walk}) = 1 - P((v_i,\ v_F) \in T_{walk}) \tag{6}$$

$$\log P\left(\left(v_i,\ v_j\right) \in T_{walk}\right) + \beta \cdot E_{v_F}\left(P\left(v_i,\ v_j\right) \notin T_{walk}\right) \tag{7}$$

Then, we use the generated graph-node vector to concatenate the text vector that is pre-trained with *gensim* (https://radimrehurek.com/gensim/). With this mechanism,

we can integrate the graph feature of a node with its text feature. Finally, we take the concatenated vector as input and measure the graph-node similarity based on the cosine similarity, as shown in Eq. (8).

$$sim(v_i, v_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|} \tag{8}$$

4.3 Textual Relevance

In addition to the graph-node similarity between nodes in the graph, we also consider the textual relevance that can reflect the similarity of the documents between companies crawled from web encyclopedia.

In Doc2vec [17], each sentence is represented by a unique vector which is expressed as a column of matrix D. Each word is also represented by a unique vector which is represented by a certain column of the matrix W. The word vector corresponding to the input word and the sentence vector corresponding to the sentence are used as the input of the input layer. The vector of the sentence and the word vector of the sample are added to form a new vector X. Then, we can use this vector X to predict the predicted words in the window. Doc2vec differs from Word2vec in that Doc2vec introduces a new sentence vector to the input layer. Paragraph vectors can also be regarded as another kind of word vectors. As a result, we use the concatenation of the vector representing three contextual words to predict the fourth word. The paragraph vector representing the paragraph topic is used to avoid missing information in the current context.

Based on the above representation of documents, we further employ the cosine similarity to compute the textual relevance between two companies. Basically, for a given company, the companies with high similarity scores will be regarded as the main competitors of the given company. In our implementation, we return the top-k companies as potential competitors for a given company.

5 Performance Evaluation

We conduct experiments on data sets crawled from Wikipedia and DBPedia to evaluate the performance of the proposed graph-embedding-based competitor mining algorithm. All the experiments are performed on a computer with an Intel Core i7-7700 3.6 GHz CPU, 16 GB DRAM, and a 512 GB SSD. All algorithms are implemented by Python 3.5.

5.1 Settings

Data Set. The data set was crawled from Wikipedia and DBPedia. We especially focus on the competitive relationships in Wikipedia and DBPedia. As a result, we prepare a data set of companies that contain 2616 companies from different industries. As shown in Table 1, the heterogeneous graph constructed from the data set has 12692 entity nodes, and 32898 edges, in which 16403 are homogenous edges, and the rest are heterogeneous edges.

Table 1. Characteristics of the graph constructed from the data set

Dataset	Company Graphs
|V|	12692
|E|	32898
$\|E_{ho}\|$	16403
$\|E_{he}\|$	16495

Among the extracted entities from Wikipedia and DBPedia, we choose 50 companies that are listed in Fortune 500 to measure the effectiveness of the graph embedding based competitor mining algorithm. One problem is the lack of annotated ground truth. The best way is to conduct a substantial survey over major companies but this is time-consuming. Therefore, we take the evaluation method used in the literature [19], which uses the recommendation results of Google search when we input a specific company name as the search criteria.

Compared Methods. In our experiments, we mainly compare our graph embedding approach with four previous algorithms, as described below.

(1) *Pathsim* [18]. This method computes node similarity in a heterogeneous graph by meta-paths. We use two kinds of meta-paths to extract competitive relationships from the data set, which are "C(company) - I(industry)-C(company)" and "C(Company) - P(Product) - PT(Product Type) - P(Product) - C(Company)". If there is a meta-path connected to a given company in the graph which is within the above two kinds of meta-paths, we output the company entities on the meta-path as the competitors of the company.
(2) *Doc2vec* [17]. This method is a text mining algorithm, which learns the text vector of a company based on the text corpus crawled from Wikipedia. We use *gensim* to train document vectors and set the dimension number to 200 by default.
(3) *Metapath2vec* [12]. This is also a heterogeneous graph embedding method which adopts meta-path-based random walks to construct the heterogeneous neighborhood of a node. Then, it uses a skip-gram model to perform node embeddings. We use this model to train node embeddings and set the dimension number to 200.
(4) *Hin2vec* [20]. This method applies the neural network model to carry out multiple prediction training tasks for a target set of relationships. It finally learns the latent vectors of nodes and meta-paths in a heterogeneous information network.

Parameters Setting. The settings of the parameters in the graph embedding are as follows: $r = 30$, $L_{max} = 30$, $k = 10$, *and* $d = 200$.

5.2 Results

Figures 2, 3, and 4 show the comparison of precision, recall, and F1-measure, respectively, for the five methods. The X-axis represents the top-n competitors returned by

each method. As companies mainly focus on their major competitors in the market, we measure the performance of each method for extracting top-5, top-10, and top-15 competitors.

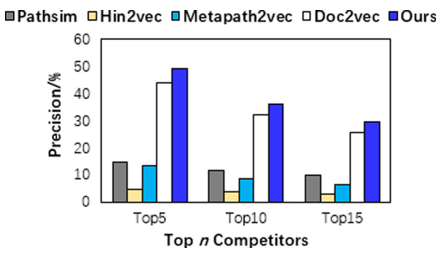


Fig. 2. Precision comparison of the five methods

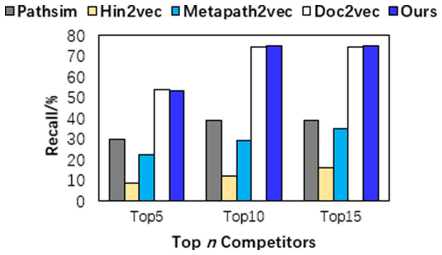


Fig. 3. Recall comparison of the five methods

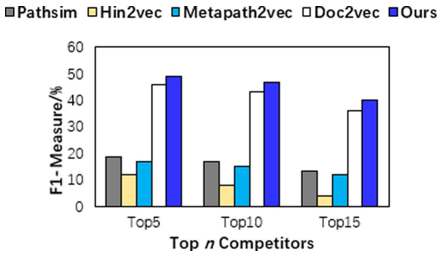
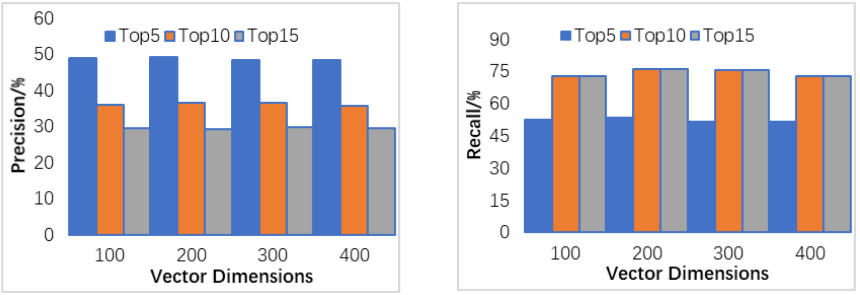


Fig. 4. F1-measure comparison of the five methods

We can see that all figures report a very similar trend. Pathsim, Metapath2vec, and Hin2vec all get the worst performance in all experiments, indicating that it is not a good choice to only consider the graph structure in competitor mining. In addition, the text-only method Doc2vec performs better than Pathsim, Metapath2vec, and Hin2vec, meaning that textual information is more important for competitor mining from web encyclopedia. This is because the documents in Wikipedia and DBpedia are semi-structured and with good quality. Our method that combines the graph embedding and textual relevance outperforms all the other four methods. This is mainly due to the combination of graph embedding and textual relevance. Thus, we can conclude that it is better to consider both graph structural information and textual relevance for mining competitors from web encyclopedia.

Next, we vary the dimension size to measure the scalability of our method concerning the dimension size. The results in terms of precision and recall are shown in Fig. 5. We can see that the precision of our method is relatively stable when the dimension size varies from 100 to 400. Therefore, our method can be extended to larger documents that involve more dimensions. Note that the documents in web encyclopedia like Wikipedia and DBPedia are generally smaller than free web pages, e.g., news web pages. Thus, we believe that our method can also be applied to free web pages in case we devise an effective approach to remove the incredible information from free web pages.

Finally, we measure the effect of parameters on the performance of our method. The results are shown in Fig. 6, where four parameters including α, k, L, and w are measured.



(a) Precision (b) Recall

Fig. 5. Scalability of our method w.r.t. dimension size

The parameter α defines the probability of node traversing during the random walk. We can see that our method achieves the best precision when it is set to 0.4. The parameter k represents the number of random walks. Our experimental result shows that higher k will not always result in the improvement of the performance. Instead, a middle value, which is 30 in our experiment, is appropriate for our method. This is also applicable to the setting of the parameter L, which is the length of a random walk. The parameter w represents the window size in the SkipGram model. Our experiment shows that it is better to use a small window size rather than a large one, indicating only the words nearby are meaningful in the training of graph embedding vectors.

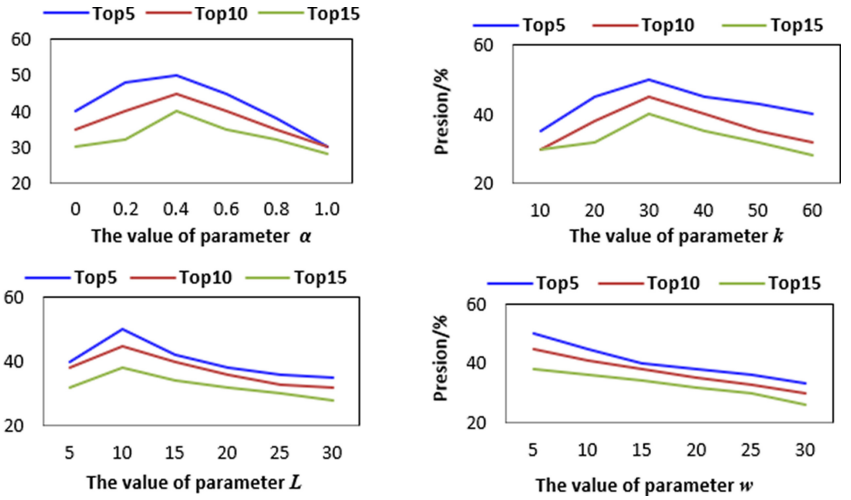


Fig. 6. Impact of the parameters on the performance

6 Conclusions

In this paper, we study the problem of mining competitive relationships by learning heterogeneous graphs constructed from web encyclopedia including Wikipedia and DBPedia. We develop a general framework for mining competitors from web encyclopedia. Particularly, we use the similarity among graph nodes to measure competitive relationships and propose to extract competitors based on graph-node similarity and textual relevance. We evaluate the proposed method on data sets crawled from Wikipedia and DBpedia, and compare our algorithm with four existing algorithms. The experimental results suggest the effectiveness of our proposal.

Some future works are worth further investigating. First, companies may has different competitive relationships [21], which need to be further distinguished. Second, we will study the credibility evaluation [22] of Wikipedia information to enhance the validity of the detected competitive relationships.

Acknowledgement. This study is supported by the National Key Research and Development Program of China (2018YFB0704404) and the National Science Foundation of China (61672479). Peiquan Jin is the corresponding author.

References

1. Bao, S., Li, R., Yu, Y., Cao, Y.: Competitor Mining with the Web. IEEE Trans. Knowl. Data Eng. **20**(10), 1297–1310 (2008)
2. Bondarenko, A., et al.: Comparative web search questions. WSDM, 52–60 (2020)
3. Zhao, J., Jin, P.: Conceptual modeling for competitive intelligence hiding in the internet. J. Softw. **5**(4), 378–386 (2010)
4. Zhao, J., Jin, P.: Towards the extraction of intelligence about competitor from the web. In: Lytras, M.D., et al. (eds.) Visioning and Engineering the Knowledge Society. A Web Science Perspective. Lecture Notes in Computer Science, vol. 5736, pp. 118–127. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04754-1_13
5. Chen, X., Wu, Y.: Web mining from competitors' websites. In: KDD, pp. 550–555 (2005)
6. Li, S., Lin, C., Song, Y., Li, Z.: Comparable Entity mining from comparative questions. In: ACL, pp. 650–658 (2010)
7. Ruan, T., Xue, L., Wang, H., Pan, J.: Bootstrapping yahoo! finance by wikipedia for competitor mining. In: Qi, G., Kozaki, K., Pan, J., Yu, S. (eds.) Semantic Technology. Lecture Notes in Computer Science, vol. 9544, pp. 108–126. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-31676-5_8
8. Lange, D., Böhm, C., Naumann, F.: Extracting Structured Information from Wikipedia Articles to Populate Infoboxes. CIKM, pp. 1661–1664 (2010)
9. Haidar-Ahmad, L., Zouaq, A., Gagnon, M.: Automatic extraction of axioms from wikipedia using SPARQL. In: Sack, H., et al. (eds.) The Semantic Web. ESWC 2016. Lecture Notes in Computer Science, vol. 9989, pp. 60–64. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47602-5_13
10. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. KDD, 701–710 (2014)
11. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. KDD, 855–864 (2016)

12. Dong, Y., Chawla, N., Swami, A.: metapath2vec: scalable representation learning for heterogeneous networks. KDD, 135–144 (2017)
13. Haghighat, M., Li, J.: Toward fast regex pattern matching using simple patterns. In: ICPADS, pp. 662–670 (2018)
14. Hill, B., Shaw, A.: Consider the redirect: a missing dimension of wikipedia research. OpenSym **28**(1–28), 4 (2014)
15. Tamir, R.: A random walk through human associations. In: ICDM, pp. 442–449 (2005)
16. Pickhardt, R., et al.: A generalized language model as the combination of skipped n-grams and modified Kneser Ney smoothing. In: ACL, pp. 1145–1154 (2014)
17. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196 (2014)
18. Sun, Y., Han, J., Yan, X., Yu, P., Wu, T.: PathSim: meta path-based top-K similarity search in heterogeneous information networks. PVLDB **4**(11), 992–1003 (2011)
19. Ni, C., Liu, K., Torzec, N.: Layered graph embedding for entity recommendation using wikipedia in the yahoo! knowledge graph. In: WWW, pp. 811–818 (2020)
20. Fu, T., Lee, W., Lei, Z.: HIN2Vec: explore meta-paths in heterogeneous information networks for representation learning. In: CIKM, pp. 1786–1806 (2017)
21. Zhao, J., Jin, P., Liu, Y.: Business relations in the web: semantics and a case study. J. Softw. **5**(8), 826–833 (2010)
22. Zhao, J., Jin, P.: Extraction and credibility evaluation of web-based competitive intelligence. J. Softw. **6**(8), 1513–1520 (2011)