EncyCatalogRec: catalog recommendation for encyclopedia article completion[*]

Wei-ming LU[†‡], Jia-hui LIU, Wei XU, Peng WANG, Bao-gang WEI[^]

College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

[†]E-mail: luwm@zju.edu.cn

Abstract: Online encyclopedias such as Wikipedia provide a large and growing number of articles on many topics. However, the content of many articles is still far from complete. In this paper, we propose EncyCatalogRec, a system to help generate a more comprehensive article by recommending catalogs. First, we represent articles and catalog items as embedding vectors, and obtain similar articles via the locality sensitive hashing technology, where the items of these articles are considered as the candidate items. Then a relation graph is built from the articles and the candidate items. This is further transformed into a product graph. So, the recommendation problem is changed to a transductive learning problem in the product graph. Finally, the recommended items are sorted by the learning-to-rank technology. Experimental results demonstrate that our approach achieves state-of-the-art performance on catalog recommendation in both warm- and cold-start scenarios. We have validated our approach by a case study.

Key words: Catalog recommendation; Encyclopedia article completion; Product graph; Transductive learning

1 Introduction

Online encyclopedias such as Wikipedia and Baidu Baike (one of the largest collaborative encyclopedias in China, which claims that it contains about 14.5 million articles) provide articles on many topics. However, there are still many entities that have not been authored, which are marked as red-linked articles (https://en.wikipedia.org/wiki/Wikipedia: Red_link), and the number continues to increase.

Although some articles have been edited several times, the content is still far from complete. Taking Fig. 1 as an example, there are several articles about acid at different levels of details in Wikipedia. For example, "sulfuric acid" has more comprehensive content than other entities, while "fluoroboric acid" and "bromic acid" have very incomplete content with fewer catalog items to describe the entities. Some entities such as "tungstic acid" and "titanic acid" even do not have any catalog to organize the content. There are only 2.9 catalog items on average (after removing uninformative catalog items such as "see also," "references," and "external links") for the total 38 articles in the "mineral acids" category (https://en.wikipedia.org/wiki/Category:Mineral_ acids), and 13 of them have no catalogs. Obviously, providing various catalog items for articles is the

Fig. 1 Wikipedia articles with different levels of details on the content, where some of them such as "fluoroboric acid" and "bromic acid" have very incomplete content with fewer catalogs to describe the entities

key for encyclopedia article completion. Therefore, in this study, we would like to help generate a more comprehensive encyclopedia article via catalog recommendation.

Several approaches have been proposed to provide suggestions for populating Wikipedia articles (Fetahu et al., 2015; Wulczyn et al., 2016) or even automatically generating the articles (Sauper and Barzilay, 2009; Banerjee and Mitra, 2015a, 2015b, 2016). However, to the best of our knowledge, all the above-mentioned approaches focused on content generation by summarization techniques including extractive approaches (Sauper and Barzilay, 2009) and abstractive approaches (Banerjee and Mitra, 2015a, 2015b, 2016), simplifying catalog generation for new articles. For example, WikiKreator (Banerjee and

Mitra, 2015b) assumes that articles in the same category have the same catalogs, and it just takes the titles of the top 10 most frequent sections as the catalog. Sauper and Barzilay (2009) and Banerjee and Mitra (2016) applied repeated bisection clustering (Zhao et al., 2005) on section content and section headings, respectively. Then the most frequent titles within each cluster were considered as the catalog.

In this study, we propose EncyCatalogRec, a system to help generate a more comprehensive article in encyclopedias by recommending catalogs. Besides the red-linked entities, we want to recommend catalogs for the entities that are not even mentioned in encyclopedias such as Baidu Baike and Wikipedia. If we want to generate a catalog for entity e from scratch, EncyCatalogRec will first find the existing

encyclopedia articles that are semantically similar or relevant to entity e, and then recommends a comprehensive and orderly catalog for entity e based on the structure of these similar articles.

In addition, our catalog recommendation approach is orthogonal to the section content generation approaches. So, summarization techniques used in Sauper and Barzilay (2009) and Banerjee and Mitra (2015a, 2015b, 2016) can also be applied in our system.

2 Related work

Wikipedia articles are a valuable source of information for knowledge base construction (Suchanek et al., 2007; Bizer et al., 2009; Hoffart et al., 2013) and semantic relatedness computation (Strube and Ponzetto, 2006). However, not all Wikipedia articles are comprehensive.

To address this issue, several approaches have been proposed either to provide suggestions for populating Wikipedia articles (Fetahu et al., 2015; Wulczyn et al., 2016) or to automatically generate the articles (Sauper and Barzilay, 2009; Banerjee and Mitra, 2015a, 2015b, 2016).

For instance, Sauper and Barzilay (2009) generated a content template for each domain by clustering all section headings from articles in the domain-specific categories, and then formulated content selection for all sections as a structured classification problem. WikiKreator (Banerjee and Mitra, 2015b) employs a topic model based text classifier to assign web excerpts into various sections of an article, and then formulates an integer linear programming (ILP) problem to generate abstractive summaries for each section with the objective of maximizing linguistic quality and information content. However, they also relied on the Wikipedia categories to obtain the domain-specific articles, and used the 10 most frequent sections in each category as the catalog for the new article. WikiWrite (Banerjee and Mitra, 2016) obtained semantically similar articles by representing entities using a paragraph vector model (Le and Mikolov, 2014), and then content from similar articles was used to train multi-class classifiers that can assign web-retrieved content on the red-linked entity to relevant sections of the article. Finally, they used a two-step ILP-based paraphrastic summarization technique to generate short, concise, and paraphrased summaries for each section.

Other approaches try to provide suggestions for populating Wikipedia articles. For instance, Wulczyn et al. (2016) proposed an end-to-end system for recommending articles that exist in one language but are missing in another. They identified missing articles, ranked the missing articles according to their importance, and recommended important missing articles to editors based on their interests. Fetahu et al. (2015) proposed a two-stage supervised approach for suggesting news articles for Wikipedia entities to improve news coverage in Wikipedia, and reduced the lag of newsworthy references.

Reinanda et al. (2015) also recommended entity aspects. However, the aspects were mined from entity-oriented logs. Tanaka et al. (2010) proposed a method for acquiring a set of high-quality query patterns to retrieve documents containing important information about an entity. Wagstaff et al. (2016) tried to automatically construct a Mars target encyclopedia by applying information extraction methods to planetary science publications, but they focused only on entity property extraction.

3 Proposed approach

In this section, we first formulate the encyclopedia catalog recommendation problem, and then elaborate it in two phases: catalog item recommendation and item ranking.

3.1 Problem definition

Our goal is to recommend comprehensive catalogs for the entities that have incomplete content or have no corresponding articles. Let $A = \{a_i\}_{i=1}^{N}$ denote a set of encyclopedia articles, where a_i includes a title t_i and a catalog c_i of the article. Here, catalog c_i is a sequence of catalog items, which can be denoted as $\{c_{ij}\}_{j=1}^{K_i}$, and c_{ij} is the jth catalog item in catalog c_i with K_i the number of catalog items in c_i.
Definition 1 (Catalog recommendation problem) Given a query entity q and a set of encyclopedia articles A, the catalog recommendation task aims to recommend a catalog c_q for q, where c_q consists of K suitable, comprehensive, and orderly catalog items $\{c_{qj}\}_{j=1}^{K}$.

Therefore, our approach consists of two phases, catalog item recommendation and catalog item ranking. The first phase is to generate the comprehensive

catalog items, and the second phase is to rank the catalog items for readability. These two phases will be illustrated in the following two subsections.

3.2 Catalog item recommendation

If we consider the articles and their corresponding catalog items as two types of nodes in a bipartite graph, we can formulate catalog item recommendation as the bipartite edge prediction problem (Liu and Yang, 2015).

Definition 2 (Bipartite edge prediction problem) Given two graphs $\mathcal{G}$ and $\mathcal{H}$, generate a complete bipartite graph $\mathcal{B}$ with nodes $V_\mathcal{B} = \{V_\mathcal{G}, V_\mathcal{H}\}$ and edges $E_\mathcal{B} = V_\mathcal{G} \times V_\mathcal{H}$, where some edges $E_\mathcal{B}^\mathrm{l}$ are labeled with $\mathcal{T} = \{y_{ij} \in \mathcal{Y} | (i,j) \in E_\mathcal{B}\}$, and then bipartite edge prediction aims to predict the labels of other edges $E_\mathcal{B}^\mathrm{u} = E_\mathcal{B}/E_\mathcal{B}^\mathrm{l}$.

In our catalog item recommendation problem, $\mathcal{G} = \{V_A \cup q\}$ and $\mathcal{H} = \cup_i \{c_{ij} | a_i \in A, j = 1, 2, ..., |c_i|\}$, where V_A is the set of articles A and c_{ij} is the jth item in catalog c_i of article $a_i \in A$. Since some encyclopedia articles already have some catalog items, these relationships between articles and items can be considered as the labeled edges in $\mathcal{B}$. Therefore, in a cold-start task where all items could be recommended to the query entity q, we have $E_\mathcal{B}^\mathrm{l} = \{(a_i, c_{ij}) | a_i \in V_A, j = 1, 2, ..., |c_i|\}$ and $E_\mathcal{B}^\mathrm{u} = \{(q, c_{ij}) | c_{ij} \in \mathcal{H}\}$.

Actually, the bipartite edge prediction problem was addressed by a transductive learning strategy to propagate the labels of $E_\mathcal{B}^\mathrm{l}$ to $E_\mathcal{B}^\mathrm{u}$ (Liu and Yang, 2015). Therefore, we first learn appropriate representations for articles, catalog items, and the query entity, and then construct a relation graph $\mathcal{B}$ among articles, catalog items, and the query entity. Finally, we transform the relation graph to a product graph $\mathcal{P}$, and apply the transductive learning strategy on $\mathcal{P}$ for catalog item recommendation.

3.2.1 Article and catalog item representation

To learn appropriate representations for articles and catalog items, we first learn word embeddings by the gensim word2vec (Mikolov et al., 2013a, 2013b) package (https://radimrehurek.com/gensim/models/word2vec.html) on the set of articles A. Denote $\boldsymbol{v}_w$ as the embedding vector of word w. Thus, for article $a \in A$, suppose its title t is an N-gram term $\langle w_1, w_2, ..., w_N \rangle$. We can obtain its semantic

representation $\boldsymbol{v}_a$ as $(\boldsymbol{v}_{w_1} + \boldsymbol{v}_{w_2} + ... + \boldsymbol{v}_{w_N})/N$. Similarly, we can obtain the semantic representation $\boldsymbol{v}_{c_{ij}} = (\boldsymbol{v}_{w_1} + \boldsymbol{v}_{w_2} + ... + \boldsymbol{v}_{w_M})/M$ for the catalog item $c_{ij} = < w_1, w_2, ..., w_M >$, which is an M-gram term.

3.2.2 Relation graph construction

We build the relation graph $\mathcal{B}$ according to the structure of encyclopedia articles in A. Fig. 2 is an example of the relation graph.
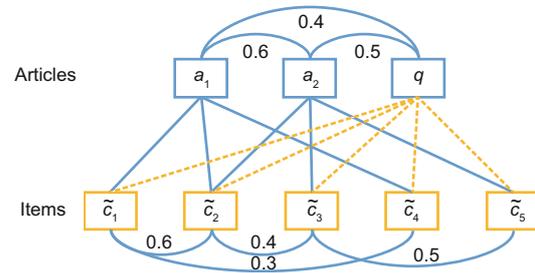

Fig. 2 Relation graph for articles and catalog items

In the figure, q is the query entity for which we want to recommend a catalog, a_1 and a_2 are two semantically similar encyclopedia articles in A, and $\{\widetilde{c}_i\}_{i=1}^5$ is a set of catalog item clusters related to a_1 and a_2. Because encyclopedia articles are written collaboratively, the catalog items may have different surface names but they are actually the same. For example, brief introduction, abstract, and overview are indeed semantically similar catalog items, so these items should be grouped as a whole for recommendation. Therefore, we cluster catalog items into semantic groups by repeated bisection clustering (Zhao and Karypis, 2002) using the representation of catalog items. Finally, we retain only clusters that have an intra-cluster similarity of at least 0.8, and set the most common catalog item in each cluster as the representative catalog item of this cluster. We denote the ith cluster as $\widetilde{c}_i \in \mathcal{C}$, and use the average vector $\boldsymbol{v}_{\widetilde{c}_i} = \sum_{c_{ij} \in \widetilde{c}_i} \boldsymbol{v}_{c_{ij}} / |\widetilde{c}_i|$ to represent the cluster.

Formally, given a query entity q with an N-gram term $\langle w_1, w_2, ..., w_N \rangle$, we calculate its semantic representation $\boldsymbol{v}_q = (\boldsymbol{v}_{w_1} + \boldsymbol{v}_{w_2} + ... + \boldsymbol{v}_{w_N})/N$, and then obtain the top M similar articles $A_q(M) \in A$ by sorting the cosine similarities between $\boldsymbol{v}_q$ and $\boldsymbol{v}_{a_i}$ $(a_i \in A)$. Then we build a relation graph $\mathcal{B}$ using $A_q(M)$ with their corresponding catalog item clusters $c(A_q(M))$, where $\mathcal{G} = \{A_q(M) \cup q\}$ and $\mathcal{H} = c(A_q(M))$. The weight between nodes in $\mathcal{G}$

is calculated by the cosine similarity $w(a_i, a_j) = \cos(\boldsymbol{v}_{a_i}, \boldsymbol{v}_{a_j})$ or $w(q, a_i) = \cos(\boldsymbol{v}_q, \boldsymbol{v}_{a_i})$, and the weight between two catalog item clusters $\widetilde{c}_i$ and $\widetilde{c}_j$ in $\mathcal{H}$ is calculated by the co-occurrence of catalog items. That is to say, $w(\widetilde{c}_i, \widetilde{c}_j) = |c_{mn} \in \widetilde{c}_i, c_{pq} \in \widetilde{c}_j, c_{mn} \in a.c, c_{pq} \in a.c, a \in A|$. If $a_i \in A_q(M)$ contains a catalog item in $\widetilde{c}_j \in c(A_q(M))$, then there is an edge between a_i and $\widetilde{c}_j$, which can be considered as the labeled edge with a weight equaling 1 in $E_{\mathcal{B}}$. Otherwise, the weight equals 0. We would like to predict whether $\widetilde{c}_i \in c(A_q(M))$ is suitable to be a catalog item for q.

3.2.3 Product graph construction

To propagate the knowledge from $E_{\mathcal{B}}^l$ to $E_{\mathcal{B}}^u$, we transform the relation graph $\mathcal{B}$ to a product graph $\mathcal{P}$, so the problem of predicting weights of edges in the relation graph is transformed to predicting weights of nodes in the product graph.

The product graph $\mathcal{P}$ from $\mathcal{G}$ and $\mathcal{H}$, denoted as $\mathcal{P} = \mathcal{G} \circ \mathcal{H}$, is also a graph with nodes $V_{\mathcal{P}} = V_{\mathcal{G} \circ \mathcal{H}} = E_{\mathcal{B}} = V_{\mathcal{G}} \times V_{\mathcal{H}}$, and edges $E_{\mathcal{P}} = E_{\mathcal{G} \circ \mathcal{H}} \subseteq V_{\mathcal{P}} \times V_{\mathcal{P}}$. Then the labeled edges $E_{\mathcal{B}}^l$ and unlabeled edges $E_{\mathcal{B}}^u$ are transformed into $V_{\mathcal{P}}^l$ and $V_{\mathcal{P}}^u$, respectively. That is to say, the weight of a node $(i \in V_{\mathcal{G}}, j \in V_{\mathcal{H}}) \in V_{\mathcal{P}}^l$ equals y_{ij} (the weight of $(i, j) \in E_{\mathcal{B}}^l$), and we would like to predict the weight of nodes in $V_{\mathcal{P}}^u$. The weight of an edge $((i, j), (i', j')) \in E_{\mathcal{P}}$ is the similarity between two edges (i, j) and (i', j') in $\mathcal{B}$, which can be calculated as the weight between two nodes i and i' in $\mathcal{G}$ multiplied by the weight between two nodes j and j' in $\mathcal{H}$.

Through this transformation, the catalog item prediction problem is transformed to predicting weights of nodes in $V_{\mathcal{P}}^u$ with some labeled nodes $V_{\mathcal{P}}^l$ on the product graph $\mathcal{P}$. Fig. 3 is an example of the product graph transformed from the relation graph
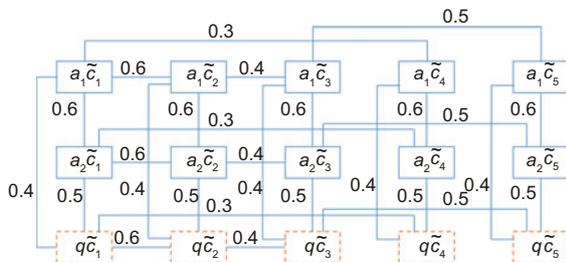

Fig. 3 Product graph generated from the relation graph in Fig. 2, where nodes with dotted boxes are unlabeled nodes (References to color refer to the online version of this figure)

in Fig. 2.

In Fig. 3, there are two types of nodes, marked in blue and orange, where nodes in blue are labeled nodes $V_{\mathcal{P}}^l$ and nodes in orange are unlabeled nodes $V_{\mathcal{P}}^u$. For edge weight calculation, we take the edges $(a_1\widetilde{c}_1, a_1\widetilde{c}_4)$ and $(a_1\widetilde{c}_1, q\widetilde{c}_1)$ as examples, where $w(a_1\widetilde{c}_1, a_1\widetilde{c}_4) = w(a_1, a_1) \cdot w(\widetilde{c}_1, \widetilde{c}_4) = 0.3$ and $w(a_1\widetilde{c}_1, q\widetilde{c}_1) = w(a_1, q) \cdot w(\widetilde{c}_1, \widetilde{c}_1) = 0.4$.

3.2.4 Recommendation over the product graph

After product graph construction, we apply a transductive learning strategy to predict the weights of nodes in $V_{\mathcal{P}}^u$ such as $q\widetilde{c}_1$ and $q\widetilde{c}_2$ in Fig. 3.

Let $m = |V_{\mathcal{G}}|$ and $n = |V_{\mathcal{H}}|$, and then we denote $\boldsymbol{r} \in \mathbb{R}^{mn}$ as the estimation vector, where r_{ij} is the estimation value on node $(a_i\widetilde{c}_j) \in V_{\mathcal{P}}$. Then we can define the transductive learning objective function as $\min_{\boldsymbol{r}} l(\boldsymbol{r}, \mathcal{T}) + \lambda \cdot \boldsymbol{r}^{\mathrm{T}}[\kappa(\boldsymbol{W})]^{-1}\boldsymbol{r}$ subject to $\boldsymbol{r}^{\mathrm{T}}\boldsymbol{r} = 1$, where $l(\boldsymbol{r}, \mathcal{T})$ is the loss function measuring the discrepancy between r and ground truth $\mathcal{T}$, $\boldsymbol{W} \in \mathbb{R}^{mn \times mn}$ is the edge weight matrix of $\mathcal{P}$ where $W_{(i,j)(i',j')}$ is the weight of edge $((i, j), (i', j')) \in E_{\mathcal{P}}$ and equals $w(ij, i'j')$, and $\kappa : \mathbb{R}^{mn \times mn} \to \mathbb{R}^{mn \times mn}$ maps the edge weight matrix to a kernel matrix related to graph transduction. There are several kernel mapping schemes such as exponential kernel, fixed-step random walk, von-Neumann kernel, and sigmoid kernel (Liu and Yang, 2015). In this study, we use the exponential kernel $\kappa(z) = \exp(z)$, since it achieved the best performance according to Liu and Yang (2015), and use a Gaussian function to initialize $\boldsymbol{r}$ for the gradient descent algorithm:

$$\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}_{mn}, \kappa(\boldsymbol{W})), \tag{1}$$

where $\boldsymbol{0}_{mn}$ is an all-zero vector in $\mathbb{R}^{mn}$.

Finally, we obtain the top K catalog item clusters $\{\widetilde{c}_i\}_{i=1}^K$ for query entity q by sorting $\boldsymbol{r}$ in descending order. The representative item in each cluster can form a catalog together.

3.3 Catalog item ranking

After obtaining the top K catalog item clusters, we should sort them according to the prerequisite relations to form a catalog for readability. Prerequisite relations essentially can be considered as the dependency among sections in a Wikipedia article, and are crucial for people to learn the knowledge in articles. In this study, we apply learning-to-rank technologies

to sort the items.

Ranking support vector machine (SVM) (Joachims, 2002) is a typical learning-to-rank method. It is a variant of SVM which penalizes the number of incorrectly ranked training examples. In our ranking problem, given $\{\widetilde{c}_i\}_{i=1}^K$, we would like to learn a ranking function f to determine the prerequisite relation between two of them:

$$\widetilde{c}_i \succ \widetilde{c}_j \Leftrightarrow f(\boldsymbol{v}_{\widetilde{c}_i}) > f(\boldsymbol{v}_{\widetilde{c}_j}), \qquad (2)$$

where $\succ$ denotes the prerequisite relation between two catalog items. For example, "background" $\succ$ "application" means that people should know "background" before "application" in an article. f is a linear function $f_{\boldsymbol{w}}(\boldsymbol{v}_{\widetilde{c}_i}) = \langle \boldsymbol{w}, \boldsymbol{v}_{\widetilde{c}_i} \rangle$, where $\boldsymbol{w}$ denotes a vector of weights and $\langle\ ,\ \rangle$ stands for the inner product.

Suppose we are given a set of catalogs $\{c_i\}$, and each includes a sequence of ordered catalog items $c_i = \{c_{ij}\}_{j=1}^{K_i}$ as in article a_i. We can take these sequences as the training data in the ranking SVM. That is to say, $c_{ij} \succ c_{ik}$, when $j < k$. Therefore, we can create a training dataset S containing l labeled vectors, such as $(\boldsymbol{v}_{c_{ij}} - \boldsymbol{v}_{c_{ik}}, z_{ijk})$, where $z_{ijk} = +1$, when $j < k$; otherwise, $z_{ijk} = -1$. Then the ranking SVM formulation is as the following:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{||\boldsymbol{w}||^2}{2} + C \sum_{i, 1<j<|K_i|, 1<k<|K_i|} \xi_{ijk} \\ \text{s.t.} \quad & z_{ijk} \langle \boldsymbol{w}, \boldsymbol{v}_{c_{ij}} - \boldsymbol{v}_{c_{ik}} \rangle \geq 1 - \xi_{ijk}, \forall i, j, k, \quad (3) \\ & \xi_{ijk} \geq 0, \forall i, j, k. \end{aligned}$$

Finally, we use the SVM$^{\text{rank}}$ (Joachims, 2006) package (https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html) to train our ranking model, and predict whether $\widetilde{c}_i$ is a prerequisite for $\widetilde{c}_j$, since $\widetilde{c}_i$ and c_{ij} are in the same vector space.

4 Experiments

4.1 Experimental setup

4.1.1 Datasets

Our dataset comes from Baidu Baike. We crawled articles in Baidu Baike, and extracted the catalog for each article through XPath (https://en.wikipedia.org/wiki/XPath) pattern matching. Finally, we obtained a dataset A containing 1 653 937 articles and 1 119 213 unique catalog items.

For testing, we randomly selected articles in four categories: plant, chemical engineering, microorganism, and medical treatment. The statistics of the testing dataset T are shown in Table 1.

Table 1 Statistics of the testing dataset

Category	Number of articles	Number of items	Number of items per article
Plant	18	81	6.33
Chemical engineering	17	97	9.88
Microorganism	17	91	5.34
Medical treatment	19	46	7.21

4.1.2 Evaluation criteria

We used precision, recall, F-measure, and diversity to evaluate the models.

Given an article $a_i \in T$ with its sequential catalog items $c_i = \{c_{ij}\}_{j=1}^{K_i}$, the model would like to recommend K orderly catalog items $c'_i = \{c'_{ij}\}_{j=1}^K$ from all unique catalog items in T. Since the models may recommend not the exact but semantically similar catalog items, we defined precision, recall, and F-measure of the recommended items $\{c'_{ij}\}_{j=1}^K$ for article a_i as follows:

$$P(a_i) = \frac{\sum_{k=1}^K \max_{j=1}^{K_i} \cos(c'_{ik}, c_{ij})}{K}, \qquad (4)$$

$$R(a_i) = \frac{\sum_{k=1}^{K_i} \max_{j=1}^K \cos(c'_{ik}, c_{ij})}{K_i}, \qquad (5)$$

$$F_1(a_i) = \frac{2 \cdot P(a_i) \cdot R(a_i)}{P(a_i) + R(a_i)}, \qquad (6)$$

where $\cos(c'_{ik}, c_{ij})$ is the cosine similarity between $\boldsymbol{v}_{c'_{ik}}$ and $\boldsymbol{v}_{c_{ij}}$. Thus, the average precision, recall, and F_1 for the testing dataset T are $P = \sum_{a_i \in T} P(a_i)/|T|$, $R = \sum_{a_i \in T} R(a_i)/|T|$, and $F_1 = \sum_{a_i \in T} F_1(a_i)/|T|$, respectively.

In addition to precision, recall, and F-measure, the diversity of the recommended items is important. Here, we defined the diversity of the recommended items $\{c'_{ij}\}_{j=1}^K$ for article a_i by

$$\text{Div}(a_i) = \sum_{j=1}^{K_i} \frac{1}{K_i M'_{ij}},$$

where $M'_{ij} = \sum_{k=1}^K \delta(\arg\max_{c_{il} \in c_i} \cos(c'_{ik}, c_{il}) = c_{ij})$ is the number of recommended items that matches the catalog item $c_{ij} \in c_i$. When no matched item is found for c_{ij}, $M'_{ij} = \infty$. Finally, the average

diversity for the testing dataset T is calculated as $\text{Div} = \sum_{a_i \in T} \text{Div}(a_i)/|T|$.

4.1.3 Evaluation methodology

We tested the models on held-out article-item pairs under both cold- and warm-start scenarios. Cold-start is the task of catalog item recommendation for a new entity, while warm-start is the task of catalog completion.

In both scenarios, we first obtained the top M similar articles $A_q(M) \in A - T$ for the query entity $q \in T$. Then we constructed the relation graphs for cold- and warm-start tasks separately, as shown in Fig. 4, where the dotted links are to be predicted.

In the cold-start task, the catalog items of q are ignored, so all items should be recommended, while in the warm-start task, some catalog items of the query entity q are kept in the recommendation. Specifically, we randomly kept half of items and recommended the other items in our experiments. As shown in Fig. 4, there is no link between q and c_i in the cold-start task, while in the warm-start task, some links already exist between q and c_i.
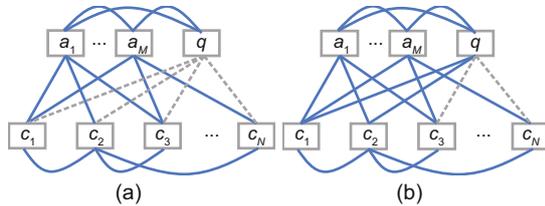

Fig. 4 Cold-start (a) and warm-start (b) scenarios, where the dotted links are to be predicted

4.1.4 Baseline methods

To evaluate our model, we compare it with the following models for catalog item recommendation:

1. Matrix factorization (MF) (Koren et al., 2009): In MF, the article latent vector $\boldsymbol{p}_{\text{a}}$ and catalog item latent vector $\boldsymbol{q}_{\text{c}}$ are learned from the article-item rating matrix by solving the following optimization problem: $\min_{\boldsymbol{p}*,\boldsymbol{q}*} \sum_{a,c}(w(a,c) - \boldsymbol{p}_{\text{a}}^{\text{T}}\boldsymbol{q}_{\text{c}})^2 + \lambda(||\boldsymbol{p}_{\text{a}}||^2 + ||\boldsymbol{q}_{\text{c}}||^2)$, where $w(a,c)$ is the recommendation value of catalog item c for article a.

2. Non-negative matrix factorization (NMF) (Luo et al., 2014): In NMF, articles and catalog items are modeled as non-negative vectors in a low-dimensional space by solving the following optimization problem: $\min_{\boldsymbol{P},\boldsymbol{Q}} ||\boldsymbol{W} - \boldsymbol{PQ}||^2 + \lambda_{\boldsymbol{P}}||\boldsymbol{P}||_{\text{F}}^2 +$

$\lambda_{\boldsymbol{Q}}||\boldsymbol{Q}||_{\text{F}}^2$ s.t. $\boldsymbol{P}, \boldsymbol{Q} \geq \boldsymbol{0}$, where $\boldsymbol{W}$ is the recommendation matrix between the articles and catalog items, and subscript "F" represents the Frobenius norm.

3. Topic-sensitive PageRank (Ts-PR) (Haveli-wala, 2002): In Ts-PR, we used a nonuniform $|A_q(M) + c(A_q(M)) + 1| \times 1$ personalization vector $\boldsymbol{q}$ to model the query entity, and then solved $\boldsymbol{P} = (1 - \alpha)\boldsymbol{MP} + \alpha\boldsymbol{q}$ iteratively, where $\boldsymbol{M}$ is the stochastic transition matrix over the relation graph, and $\alpha = 0.15$ in our experiments. The top ranked items according to $\boldsymbol{P}$ were recommended.

We denote out approach as BEP, since the core of the WikiCatalogRec is bipartite graph edge prediction.

4.1.5 Implementation details

We have 1 653 937 articles in dataset A. So, it is time consuming to obtain the top M similar articles $A_q(M) \subset A$ for a query entity $q \in T$ by sorting the cosine similarity between $\boldsymbol{v}_q$ and $\boldsymbol{v}_{a_i}(a_i \in A)$. Thus, we resorted to locality sensitive hashing (LSH) (Datar et al., 2004) to accelerate the search procedure. LSH provides efficient approximate nearest neighbor search by hashing similar high-dimensional vectors to the same buckets with a high probability. Here, open source LSHash (https://github.com/kayzhu/LSHash) was used, and the hash size was set to 10. By default, we used LSH to obtain the top M ($M = 100$) similar articles for recommending K ($K = 10$) items, and the experiments showed that LSH can accelerate the search procedure greatly.

In addition, parameter λ in the transductive learning function was set to 0.01 by default. We trained the word vectors by word2vec (Mikolov et al., 2013a, 2013b) (https://radimrehurek.com/gensim/models/word2vec.html) on dataset A, and the dimension was set to 100 for the word vectors.

4.2 Experimental results

4.2.1 Comparison to baselines

In this subsection, we carried out the experiments by recommending different numbers of catalog items with top 100 similar articles. The results of comparison to baselines in warm- and cold-start scenarios are reported in Figs. 5 and 6, respectively.

As shown in the figures, we can observe that: (1) With the increase of the number of recommended

items, the recall increased while the precision decreased for all methods; (2) BEP achieved superior precision, recall, F_1, and diversity scores in both warm- and cold-start scenarios compared with other methods; (3) The warm-start task achieved better performances than the cold-start task on all methods; (4) With the increase of the number of recommended items, F_1 scores had different trends in warm- and cold-start scenarios. The reason may be that the number of catalog items for each entity was limited. For example, the average number of items per article was only 7.178 in the testing dataset as shown in Table 1. Thus, when 50% items were provided in the warm-start scenario, the best number of the recommended items may be 3 or 4. In the cold-start scenario, we find that F_1 reached the peak
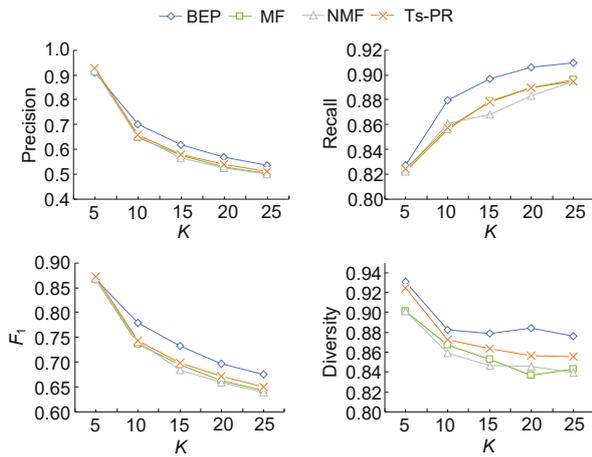
when $K = 10$ (BEP), which was close to the average number of items in the testing dataset.

4.2.2 Influence of different sizes of similar articles

The recommended catalog items were selected from the top M similar articles, so the parameter M would influence the recommendation performance. We tested our approach with M from $\{4, 5, 10, 50, 100, 150\}$, and the results are shown in Fig. 7.

From the figures, we can observe that the performance improved with the increase of the size of similar articles M, but dropped when M further increased, as too many articles may introduce noise which misleads the prediction. According to the figures, our approach achieved the best F_1 when $M = 10$ for the cold-start scenario and $M = 5$ for the warm-start scenario.

4.2.3 Influence of initial items

In the warm-start scenario, the entity already has some catalog items, but is not complete, so we should recommend some other items. In the cold-start scenario, all catalog items should be recommended for the entity. Thus, we have investigated the influence of the initial items for recommendation. The results are shown in Fig. 8, where in the

Fig. 5   Results of the warm-start scenario for recommending different numbers of catalog items

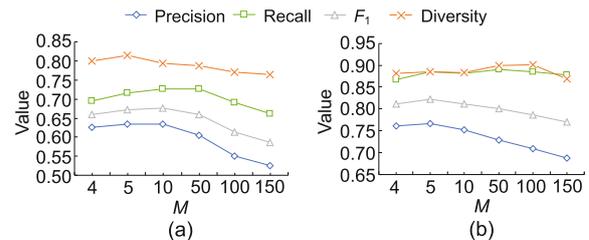**Fig. 7   Influence of different sizes of similar articles for the cold-start (a) and warm-start (b) scenarios**

Fig. 6   Results of the cold-start scenario for recommending different numbers of catalog items
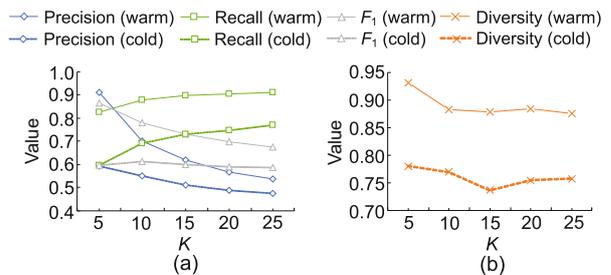
Fig. 8   Performance comparison between warm- and cold-start scenarios: (a) precision, recall, and F_1; (b) diversity

warm-start scenario, we randomly kept half of the items for the entity.

From the figures, we see that the initial items can improve the recommendation performance on all measures including precision, recall, F_1, and diversity.

4.2.4 Influence of locality sensitive hashing

To quickly retrieve similar entities, we used LSH to accelerate the search procedure. In this subsection, we investigate the influence of LSH on our approach, and the results are shown in Figs. 9 and 10, where "w/o LSH" denotes the approaches retrieving similar entities without using LSH, and "LSH" denotes the approaches using LSH to accelerate the search procedure.

From the figures, we can see that: (1) The performance was reduced in both warm- and cold-start scenarios when using LSH for retrieving sim-

ilar entities; (2) The influence of LSH on our approach was relatively small compared to other approaches.

Table 2 shows the time of retrieving different numbers of similar articles with or without LSH. We find that it would take about 20 s to retrieve similar articles without LSH, while it took less than 1 s when using LSH.

Table 2 Runtime of retrieving similar articles with or without LSH

Condition	Runtime (s)		
	M=50	M=100	M=150
With LSH	0.148	0.176	0.188
Without LSH	19.120	24.129	24.230

4.2.5 Parameter sensitivity

Our approach involves the hyper-parameter λ in the transductive learning function. In this subsection, we examine how different choices of the parameter affect the performance of the approach. We set λ at $\{0.001, 0.01, 0.1, 1, 10\}$, and the results are shown in Fig. 11.

From the figures, we can see that when $\lambda = 0.01$, our approach achieved the best performance in terms of F_1 and diversity.
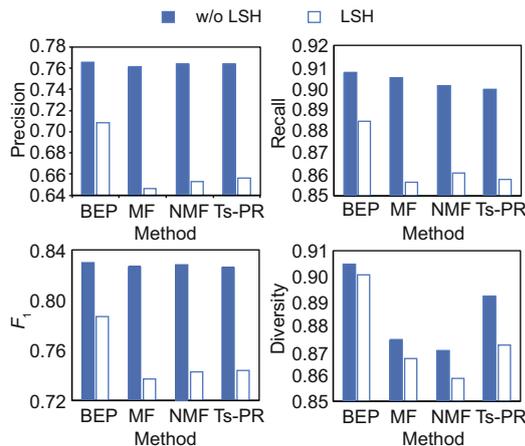

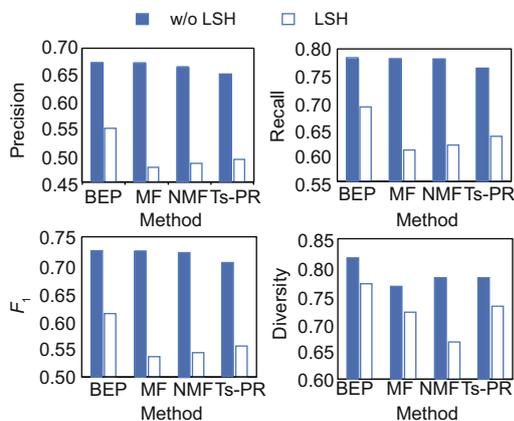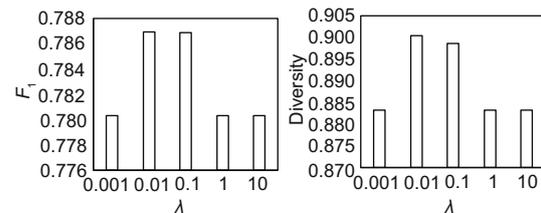**Fig. 9 Influence of LSH in the warm-start scenario**

Fig. 11 Influence of λ

4.3 Case study

In this subsection, we provide some examples of catalog item recommendation in both warm- and cold-start scenarios using our approach and other baseline approaches.

Tables 3 and 4 illustrate two examples of 智齿冠周炎 (pericoronitis) and 红枫 (red maple), respectively. In the tables, the original catalog items are listed following each entity, and the recommended items are listed for each approach.

From the tables, we can observe that: (1) BEP can recommend more correct items than other

Fig. 10 Influence of LSH in the cold-start scenario

Table 3 An example of pericoronitis for catalog item recommendation

智齿冠周炎 (pericoronitis):	病因 (cause), 临床表现 (clinical manifestation), <u>检查 (inspection)</u>, 鉴别诊断 (antidiastole), <u>治疗 (treatment)</u>, 预后 (prognosis)
Method	**Recommended items**
MF[#]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 鉴诊断别 (antidiastole), 诊断 (diagnose), 治疗 (treatment), 内容简介 (brief introduction of the content), 目录 (catalog), 版权信息 (copyright info.), 编辑推荐 (editor's choice)
NMF[#]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 鉴别诊断 (antidiastole), 治疗 (treatment), 疾病治疗 (disease treatment), 版权信息 (copyright info.), 内容简介 (brief introduction of the content), 目录 (catalog), 编辑推荐 (editor's choice)
Ts-PR[#]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 鉴别诊断 (antidiastole), 诊断 (diagnose), 疾病治疗 (disease treatment), 治疗 (treatment), 内容简介 (brief introduction of the content), 目录 (catalog), 版权信息 (copyright info.)
BEP[#]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 治疗 (treatment), 诊断 (diagnose), 鉴别诊断 (antidiastole), 并发症 (complication), 疾病治疗 (disease treatment), 预防 (precaution), 预后 (prognosis)
MF[△]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 鉴别诊断 (antidiastole), 疾病治疗 (disease treatment), 内容简介 (brief introduction of the content), 目录 (catalog), 版权信息 (copyright info.), 编辑推荐 (editor's choice), 作者简介 (about the author)
NMF[△]	病因 (cause), 临床表现 (clinical manifestation), 检查 (inspection), 诊断 (diagnose), 鉴别诊断 (antidiastole), 疾病治疗 (disease treatment), 内容简介 (brief introduction of the content), 目录 (catalog), 版权信息 (copyright info.), 作者简介 (about the author)
Ts-PR[△]	临床表现 (clinical manifestation), 检查 (inspection), 诊断 (diagnose), 鉴别诊断 (antidiastole), 治疗 (treatment), 疾病治疗 (disease treatment), 内容简介 (brief introduction of the content), 目录 (catalog), 版权信息 (copyright info.), 作者简介 (about the author)
BEP[△]	病因 (cause), 临床表现 (clinical manifestation), 治疗 (treatment), 诊断 (diagnose), 检查 (inspection), 并发症 (complication), 预防 (precaution), 预后 (prognosis), 内容简介 (brief introduction of the content), 目录 (catalog)

[#] denotes the warm-start scenario; △ denotes the cold-start scenario. Underlines denote the items selected for the warm-start scenario and wavy underlines denote the incorrectly recommended items in the results

approaches in both warm- and cold-start scenarios; (2) The approaches can recommend more correct items in the warm-start scenario than in the cold-start scenario.

Why are the wrong items such as 内容简介 (brief introduction of the content) and 版权信息 (copyright info.) recommended for the entity 智齿冠周炎 (pericoronitis), and the wrong items such as 作品鉴赏 (works appreciation) and 作品原文 (content of work) recommended for the entity 红枫 (red maple) in the baseline approaches? We inspected the similar entities to the query entity, and found that "临床专科护理技术操作规程" (Clinical Nursing Operation Specification) and "左掖梨花" (The Pear Blossom in Left Palace) are two entities in the top K similar entities of 智齿冠周炎 (pericoronitis) and 红枫 (red maple), respectively. However, "临床专科护理技术操作规程" (Clinical Nursing Operation Specification) is a book whose items are 版权信息 (copyright info.), 内容简介 (brief introduction of the content), and 目录 (catalog), while "左掖梨花" (The Pear Blossom in Left Palace) is a book of poetry whose items are 作品原文 (content of work), 注释译文 (translation and notes), 作品鉴赏 (works appreciation), and 作

者简介 (about the author). BEP is less affected by wrong similar entities, and the reason may be: (1) The product graph enables BEP to simultaneously exploit the partially labeled edges and the intrinsic structures within the entities and items; (2) Transductive learning over the product graph can leverage labeled and unlabeled edges.

5 Conclusions

In this paper, we have proposed EncyCatalogRec, a system to help generate a more comprehensive article by recommending catalogs. This is a good starting point for authors to complete articles in encyclopedias.

In EncyCatalogRec, we represented articles and catalog items by embedding vectors, and obtained the similar articles via LSH efficiently. The articles and their catalog items were used to build a relation graph, which was further transformed to a product graph. Therefore, the recommendation problem has been changed to a transductive learning problem. We also used SVM^{rank} to sort the recommended catalog items for readability. We compared our

Table 4 An example of red maple for catalog item recommendation

红枫 (red maple):	形态特征 (morphological characteristics), 生长习性 (growth habit), 分布范围 (distribution range), 主要价值 (chief value), 主要品种 (main varieties), 养护 (maintenance), 栽培技术 (cultivation technique), 红枫种植要点 (planting points for red maple), 植物文化 (plant culture)
Method	**Recommended items**
MF#	简介 (introduction), 形态特征 (morphological characteristics), 分布范围 (distribution range), 主要品种 (main varieties), 栽培技术 (cultivation technique), 植物文化 (plant culture), 信息 (information), 作品鉴赏 (works appreciation), 采集地 (collection sites), 内容简介 (brief introduction of the content)
NMF#	简介 (introduction), 形态特征 (morphological characteristics), 分布范围 (distribution range), 主要品种 (main varieties), 栽培技术 (cultivation technique), 植物文化 (plant culture), 采集地 (collection sites), 信息 (information), 地理分布 (geographical distribution), 生长习性 (growth habit)
Ts-PR#	简介 (introduction), 形态特征 (morphological characteristics), 分布范围 (distribution range), 主要品种 (main varieties), 栽培技术 (cultivation technique), 植物文化 (plant culture), 采集地 (collection sites), 信息 (information), 生长习性 (growth habit), 经济发展 (economic development)
BEP#	形态特征 (morphological characteristics), 分布范围 (distribution range), 主要品种 (main varieties), 栽培技术 (cultivation technique), 地理分布 (geographical distribution), 主要价值 (chief value), 生长习性 (growth habit), 分布情况 (distribution), 生长环境 (growth environment), 植物文化 (plant culture)
MF△	简介 (introduction), 形态特征 (morphological characteristics), 栽培技术 (cultivation technique), 采集地 (collection sites), 生长环境 (growth environment), 生长习性 (growth habit), 信息 (information), 作品原文 (content of work), 经济发展 (economic development), 自然资源 (natural resources)
NMF△	简介 (introduction), 形态特征 (morphological characteristics), 栽培技术 (cultivation technique), 采集地 (collection sites), 生长环境 (growth environment), 地理分布 (geographical distribution), 地理位置 (geographic location), 信息 (information), 作者简介 (about the author), 作品原文 (content of work)
Ts-PR△	简介 (introduction), 栽培技术 (cultivation technique), 采集地 (collection sites), 地理分布 (geographical distribution), 信息 (information), 生长习性 (growth habit), 主要价值 (chief value), 经济发展 (economic development), 作者简介 (about the author), 自然资源 (natural resources)
BEP△	形态特征 (morphological characteristics), 地理分布 (geographical distribution), 栽培技术 (cultivation technique), 采集地 (collection sites), 生长习性 (growth habit), 主要价值 (chief value), 生长环境 (growth environment), 植物文化 (plant culture), 分布情况 (distribution), 信息 (information)

# denotes the warm-start scenario; △ denotes the cold-start scenario. Underlines denote the items selected for the warm-start scenario and wavy underlines denote the incorrectly recommended items in the results

approach with other methods in both warm- and cold-start scenarios with a dataset created from Baidu Baike. The results proved that our approach can achieve the state-of-the-art performance, and can recommend useful catalog items for encyclopedia article completion.

In the future, we would like to apply neural networks such as neural collaborative filtering (He et al., 2017) to recommend the catalog items. In addition, we plan to generate content for these items automatically through summarization techniques (Gambhir and Gupta, 2017) on the relevant documents from the web.

Compliance with ethics guidelines

References

Banerjee S, Mitra P, 2015a. Filling the gaps: improving Wikipedia stubs. Proc ACM Symp on Document Engineering, p.117-120. https://doi.org/10.1145/2682571.2797073

Banerjee S, Mitra P, 2015b. WikiKreator: improving Wikipedia stubs automatically. Proc 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int Joint Conf on Natural Language Processing, p.867-877. https://doi.org/10.3115/v1/P15-1084

Banerjee S, Mitra P, 2016. WikiWrite: generating Wikipedia articles automatically. Proc 25th Int Joint Conf on Artificial Intelligence, p.2740-2746.

Bizer C, Lehmann J, Kobilarov G, et al., 2009. DBpedia—a crystallization point for the web of data. *J Web Semant*, 7(3):154-165. https://doi.org/10.1016/j.websem.2009.07.002

Datar M, Immorlica N, Indyk P, et al., 2004. Locality-sensitive hashing scheme based on p-stable distributions. Proc 20th Annual Symp on Computational Geometry, p.253-262. https://doi.org/10.1145/997817.997857

Fetahu B, Markert K, Anand A, 2015. Automated news suggestions for populating Wikipedia entity pages. Proc 24th ACM Int Conf on Information and Knowledge Management, p.323-332. https://doi.org/10.1145/2806416.2806531

Gambhir M, Gupta V, 2017. Recent automatic text summarization techniques: a survey. *Artif Intell Rev*, 47(1):1-66. https://doi.org/10.1007/s10462-016-9475-9

Haveliwala TH, 2002. Topic-sensitive PageRank. Proc 11th Int Conf on World Wide Web, p.517-526. https://doi.org/10.1145/511446.511513

He XN, Liao LZ, Zhang HW, et al., 2017. Neural collaborative filtering. Proc 26th Int Conf on World Wide Web, p.173-182. https://doi.org/10.1145/3038912.3052569

Hoffart J, Suchanek FM, Berberich K, et al., 2013. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artif Intell*, 194:28-61. https://doi.org/10.1016/j.artint.2012.06.001

Joachims T, 2002. Optimizing search engines using clickthrough data. Proc 8th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.133-142. https://doi.org/10.1145/775047.775067

Joachims T, 2006. Training linear SVMs in linear time. Proc 12th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.217-226. https://doi.org/10.1145/1150402.1150429

Koren Y, Bell R, Volinsky C, 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30-37. https://doi.org/10.1109/MC.2009.263

Le QV, Mikolov T, 2014. Distributed representations of sentences and documents. Proc 31st Int Conf on Machine Learning, p.1188-1196.

Liu HX, Yang YM, 2015. Bipartite edge prediction via transductive learning over product graphs. Proc 32nd Int Conf on Machine Learning, p.1880-1888.

Luo X, Zhou MC, Xia YN, et al., 2014. An efficient nonnegative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Trans Ind Inform*, 10(2):1273-1284. https://doi.org/10.1109/TII.2014.2308433

Mikolov T, Sutskever I, Chen K, et al., 2013a. Distributed representations of words and phrases and their compositionality. Proc 26th Int Conf on Neural Information Processing Systems, p.3111-3119.

Mikolov T, Chen K, Corrado G, et al., 2013b. Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781

Reinanda R, Meij E, de Rijke M, 2015. Mining, ranking and recommending entity aspects. Proc 38th Int ACM SI-GIR Conf on Research and Development in Information Retrieval, p.263-272. https://doi.org/10.1145/2766462.2767724

Sauper C, Barzilay R, 2009. Automatically generating Wikipedia articles: a structure-aware approach. Proc 47th Annual Meeting of the ACL and the 4th Int Joint Conf on Natural Language Processing of the AFNLP, p.208-216.

Strube M, Ponzetto SP, 2006. WikiRelate! Computing semantic relatedness using Wikipedia. Proc 21st National Conf on Artificial Intelligence, p.1419-1424.

Suchanek FM, Kasneci G, Weikum G, 2007. YAGO: a core of semantic knowledge. Proc 16th Int Conf on World Wide Web, p.697-706. https://doi.org/10.1145/1242572.1242667

Tanaka S, Okazaki N, Ishizuka M, 2010. Learning web query patterns for imitating Wikipedia articles. Proc 23rd Int Conf on Computational Linguistics, p.1229-1237.

Wagstaff KL, Riloff E, Lanza NL, et al., 2016. Creating a Mars target encyclopedia by extracting information from the planetary science literature. AAAI Workshop on Knowledge Extraction from Text, p.532-536.

Wulczyn E, West R, Zia L, et al., 2016. Growing Wikipedia across languages via recommendation. Proc 25th Int Conf on World Wide Web, p.975-985. https://doi.org/10.1145/2872427.2883077

Zhao Y, Karypis G, 2002. Evaluation of hierarchical clustering algorithms for document datasets. Proc 11th Int Conf on Information and Knowledge Management, p.515-524. https://doi.org/10.1145/584792.584877

Zhao Y, Karypis G, Fayyad U, 2005. Hierarchical clustering algorithms for document datasets. *Data Min Knowl Discov*, 10(2):141-168. https://doi.org/10.1007/s10618-005-0361-3